Apple II

IEEE-488 Interface

User's Guide

NOTICE

Apple Computer, Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

APPLE COMPUTER, INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL OR WITH RESPECT TO THE SOFTWARE DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. APPLE COMPUTER, INC. SOFTWARE IS SOLD OR LICENSED "AS IS". THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH THE BUYER. SHOULD THE PROGRAMS PROVE DEFECTIVE FOLLOWING THEIR PURCHASE, THE BUYER (AND NOT APPLE COMPUTER, INC. ITS DISTRIBUTOR, OR ITS RETAILER) ASSUMES THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL APPLE COMPUTER, INC. BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE, EVEN IF APPLE COMPUTER, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

1Ø26Ø Bandley Drive
Cupertino, California 95Ø14
(4Ø8) 996-1Ø1Ø

The word APPLE and the Apple logo are registered trademarks of APPLE COMPUTER INC.

APPLE product #A2LØØ37
(030-0197-A)

Printed in USA

Apple II

IEEE-488 Interface

User's Guide

PLEASE READ THIS MANUAL BEFORE ATTEMPTING TO INSTALL THE APPLE II IEEE-488 INTERFACE CARD IN THE APPLE COMPUTER. INCORRECT INSTALLATION COULD CAUSE PERMANENT DAMAGE TO BOTH THE INTERFACE CARD AND TO THE APPLE.

TABLE OF CONTENTS

CHAPTER 1

GETTING STARTED 1

CHAPTER 2

THE IEEE-488 INTERFACE CARD 9

CHAPTER 3

PROGRAMMING THE GPIB 19

APPENDIX C

9914 REGISTERS 49

APPENDIX D

PROGRAMMING AIDS 51

APPENDIX E

FAST TRANSFER ROUTINES 55

APPENDIX F

PROGRAMMING IN PASCAL 65

APPENDIX G

SAMPLE PROGRAMS 67

APPENDIX H

SCHEMATIC DIAGRAM 71

GLOSSARY 73

INDEX 75

CHAPTER 1
GETTING STARTED

The Apple II IEEE-488 Interface Card provides a standard, plug-compatible, input/output bus for microcomputers. It allows your Apple II to control or communicate with up to 14 external devices. The external devices may be measuring instruments, data logging devices, or one or more of many available peripheral devices. These devices must all have bus interfaces which comply with the technical, electrical and mechanical requirements of IEEE Standard 488. Approximately 6ØØ devices are currently available which use some form of this proposed standard input/output bus.

WHAT IS THE APPLE II GPIB?

The formal name for this type of bus is "IEEE Standard Digital Interface for Programmable Instrumentation, 1978". The complete document, IEEE Standard 488-1978, may be obtained from the Institute of Electrical and Electronic Engineers, Inc. The instrument bus is also known as the "IEEE-488 Interface Bus", the "General Purpose Interface Bus" (GPIB) or the "Hewlett-Packard Interface Bus" (HP-IB®). You will not need to use the IEEE Standard 488 documentation unless you need actual design information for the construction of an interface device.

The Apple II IEEE-488 Interface will be referred to as the Apple II GPIB, or just the GPIB, throughout the rest of this manual. The Apple II GPIB consists of the interface circuit card and the interface cable. When we discuss the GPIB, remember that the terms "Apple II IEEE-488 Interface Card", "IEEE-488 bus" and "HP-IB®" all refer to the same type of universal I/O bus interface and bus operation. The interface circuit card itself will be referred to as the Apple II GPIB interface card, or just as the IEEE-488 card.

The Apple II GPIB interface card is the heart of the input/output operation. The interface card plugs into an Apple II connector slot and controls all transfers over the bus. The interface card is slot-independent; in other words, the interface card may be installed in any connector slot in the Apple (except slot Ø). The interface card provides remote device control through simple BASIC commands from your program.

The interface card circuits perform the function of translating Integer BASIC or Applesoft BASIC instructions into commands understood by devices. The interface circuits include a controller which does all of the timing, control, and formatting for the instrument bus. The interface card also contains a read-only memory (ROM) that provides the control information needed by the interface controller circuit to interpret BASIC commands.

WORDS OF ADVICE

Although the electrical characteristics of the bus are clearly specified by IEEE Standard 488, the addressing conventions, input data format, and output data format can vary from one device to the next. You must carefully study the manufacturer's operating manual for each device before attempting to place the device in service on the bus. For example, the output data may be ASCII or binary. If it is binary, the least significant bit of each byte may be transmitted first or last, and so on.

There are no really good reference books available for setting up and operating a system under the IEEE-488 proposed standard. You must have a thorough command of BASIC or APPLESOFT programming and a working knowledge of DOS. If you do not, then refer to the BASIC and DOS manuals and practice the examples. You may also need to understand the relationship between ASCII, hexadecimal, octal, and binary notation. Conversion tables are included in an appendix.

A test device which can be plugged into the bus and which can be used to test the commands and data sent by your program may be advantageous. A device that can single-step the bus program is almost a necessity for complex systems. One such test device now available is the model ZT488 GPIB ANALYZER manufactured by Ziatech Corporation. Operation of the bus may also be tested by use of a Tektronix or Hewlett-Packard logic analyzer. Another useful device is the Gould Model K1ØØD/4Ø8 GPIB Analyzer.

SPECIAL SYMBOLS

This manual uses some special symbols to indicate particularly noteworthy pieces of information. If you see the symbol

it means that the following paragraph discusses some possibly unexpected Apple behavior. The symbol

means that the following paragraph contains special information that may be useful to you. Read these sections carefully.

WHAT YOU WILL NEED

The Apple II IEEE-488 Interface Card is designed for experienced users who are familiar with APPLESOFT or Integer BASIC, and DOS 3.2 or DOS 3.3. Those who wish to use the card with the Apple Pascal Operating System should be familiar with Apple Pascal, its operating system, and 65Ø2 assembly language.

The programs described in this manual, as well as the firmware used by the card, will run on an Apple II or an Apple II Plus with at least 32K memory, and at least one Disk II drive and controller.

Six items make up your Apple II IEEE-488 Interface Card package:

1. Apple II IEEE-488 Interface Card
   Part # Ø3Ø-Ø67Ø-ØØ16
2. Interconnection cable with metal clamp on it
   Part # Ø3Ø-Ø59Ø-ØØ3Ø
3. Two IEEE standard screws
4. Two non-standard screws
5. A warranty card
6. This manual

Carefully unpack all the items in your shipment. Fill out the warranty card and mail it to your Apple dealer.

INSTALLING THE INTERFACE CARD

Read this section carefully, even if you have installed Apple peripheral interface cards before. The Apple II GPIB interface card is easy to install, but it is important to install it correctly.

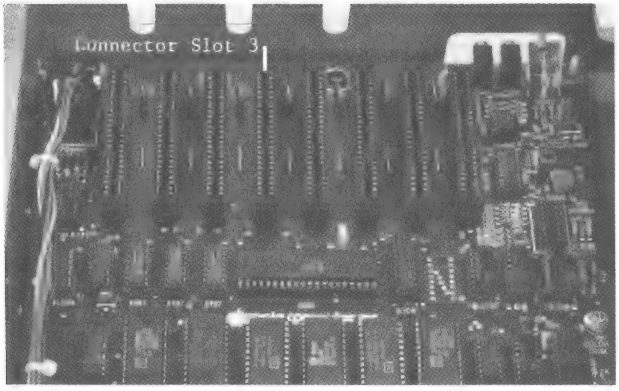
Before connecting or disconnecting anything on the Apple, turn off the power with the switch on the back left corner of the Apple case. THIS IS NECESSARY. If you try to connect or disconnect anything from the inside of your Apple when the power is on, you are likely to damage the circuits and lose anything currently stored in random access memory (RAM).

Do not unplug the Apple; just turn it off. If you unplug the Apple, you will isolate it from earth ground and leave it vulnerable to static discharges.

Remove the Apple cover by pulling up on the two back corners of the cover until the two corner fasteners pop apart. Slide the cover back until it is free of the case and lift the cover off.

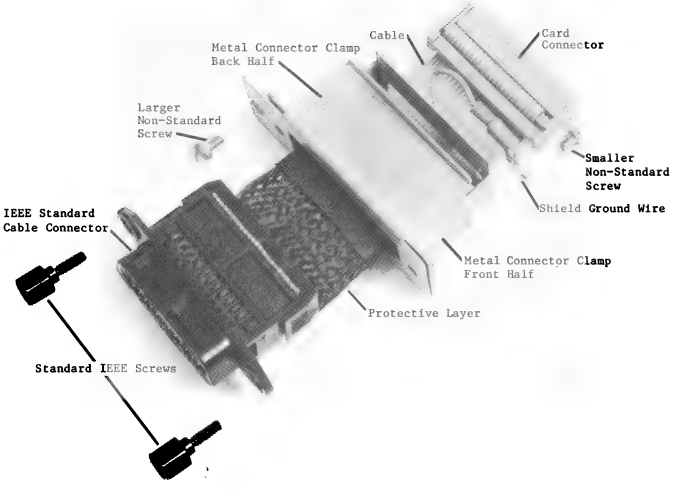
Look inside the Apple and locate the power supply case. The power supply is the rectangular metal box in the left rear corner. Touch the power supply case with one hand to discharge any static charge that may be on your clothes or body.

Along the back inside edge of the Apple you will see eight long narrow slots called "connector slots". The connector slots are numbered from Ø at the left corner to 7 at the right corner. The numbers are printed along the back edge behind the connector slots. The connector slots are shown in the following photograph.



Your disk controller card is plugged into slot 6. If you have more than two drives, a second disk controller card is plugged into slot 4 or 5. If you have an Apple Language Card, it is plugged into slot Ø. If you have a printer, the printer interface card is probably plugged into slot 1. This manual assumes that you will plug the Apple II GPIB interface card into slot 3. If you are going to plug your card into another slot, substitute that slot's number for all references to slot 3 in this manual.

The Apple IEEE interconnection cable is shown below.

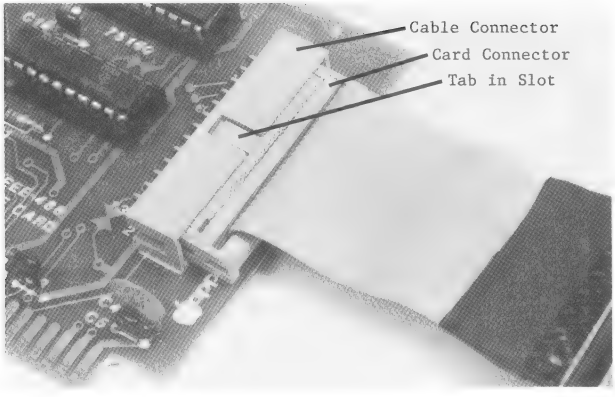


The IEEE standard cable connector is at the left. This connector is the one to which other devices will be attached. Just to the right of the IEEE connector is a two piece metal clamp. We will refer to the part of the clamp nearest the IEEE connector as the back half of the clamp. The other half of the connector clamp, the front half, has an narrow cable pathway with the cable going through it. At the right end of the cable is the interface card's connector.

When the two halves of the clamp are nestled snugly together, as they are in the photograph above, they form a rectangular enclosure with an opening for the IEEE connector on the left, an opening for the cable on the right, and two other large openings. These two openings allow the clamp to slide into one of the slots in the back of the case itself. Be sure that the two halves are together, but not too tightly; the case must fit into the openings.

The left end of the cable, by the IEEE connector, is covered by a protective layer, beneath which one of the wires, the shield ground, is severed. A new wire, attached to the IEEE connector end of the severed wire, extends from beneath the protective covering. Make sure that this wire runs parallel to the IEEE cable, and through the cable pathway in the front half of the metal connector clamp. This wire will be attached to case ground.

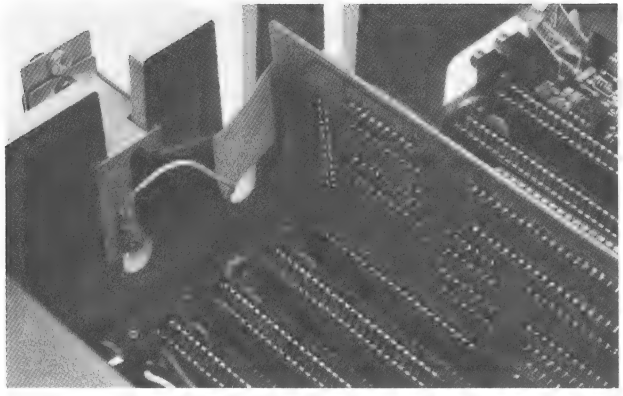
The card connector on the bus cable plugs into a connector on the end of the interface card. Hold the bus cable so that the tab on the cable's card connector is aligned with the slot on the card's connector. Press the cable's connector into the card's connector until it is firmly seated, as shown in the next photograph.



Handle the interface card as you would handle an expensive phonograph record. Grasp it only by the corners or edges, and do not touch the components or pins, especially the edge connector fingers.

Grasp the upper edge of the interface card in your right hand, and the metal connector clamp for the IEEE connector in your left. Slide the metal connector clamp into an available slot in the back of the Apple's case; if the clamp does not slide easily, pull the front and back halves a little farther apart, making the opening for the case larger. Simultaneously guide the card towards connector slot 3 with your right hand.

Insert the gold "fingers" of the interface card edge connector into slot 3, rear edge first. Gently push the front edge of the card down until it is level and firmly seated as shown in the photograph.

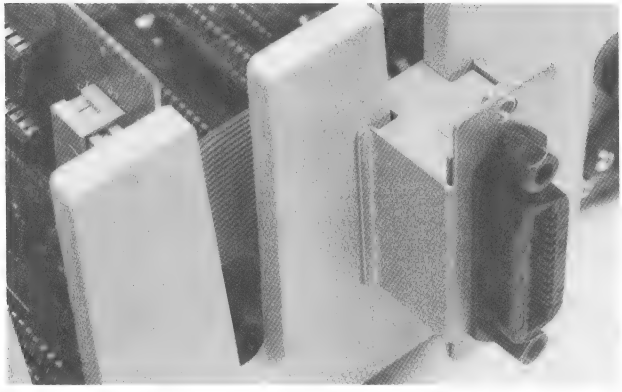


Using the smaller of the two non-standard screws, attach the shield ground wire, which should be right alongside the bus cable, to the hole in the lower part of the front half of the connector clamp. By doing so, you connect the shield ground line for the bus to the case ground of the Apple II. The line should be connected as shown in the photo above.

If your Apple II does not have a metallic interior, your shield ground will not be connected to ground.

For the final step of the installation process, turn the Apple around so the rear of the case points towards you. Make sure that the metal connector clamp rests securely in the case slot, positioned as far down as possible. Take one of the two standard IEEE screws, place it through one of the two holes in the IEEE connector so that it goes into the corresponding hole in the metal connector clamp, and tighten it. Repeat this procedure with the other IEEE screw and other connector hole.

You have just accomplished two things. You have tightened the metal connector clamp against the back of the case, preventing the connector from moving in the slot, and improving the connection between the clamp and case ground. You have also fastened the IEEE connector firmly in place so that there is no chance of putting undue strain on the cable. The connection, as seen from the rear of the Apple, appears in the next photograph.



The remaining screw may be used to suppress the radio frequency interference signals which radiate from the bus cable. If used, the screw should join the shield of the cable connected to the IEEE connector in the back of the Apple to the top hole in the front half of the metal connector clamp.

Slide the Apple case top plate in place and press down on the rear corners until the corner fasteners pop into place. The Apple II IEEE-488 Interface Bus is now installed and ready to use.

CHAPTER 2

THE IEEE-488 INTERFACE CARD

The Apple II IEEE-488 Interface Card is one of several "standardized" input/output (I/O) interfaces which have been developed since 1972 in the effort to provide a universal I/O bus.

The advantage of the Apple II GPIB is that it conforms to IEEE Standard 488 and that a single interface card lets your Apple II control and communicate with up to 14 other data gathering or processing units. It uses a commercially available standard interface cable with piggyback connectors which allow the devices on the bus to be daisy chained or star connected as shown in Figures 2-2 and 2-3. The standard cable connects to all devices designed to operate under the IEEE-488 proposed standard.

CONTROLLERS, TALKERS AND LISTENERS

The Apple II GPIB uses eight data lines and eight control lines to transfer commands and data to and from the I/O devices on the bus. A block diagram of the bus is shown in Figure 2-1. The devices on the bus can be described in terms of their primary functions of controller, talker, and listener.

Any device on the bus is designated a "talker" when its primary function is the transmission of data. An example of a talker is a digital voltmeter. If a talker transmits data over the bus continuously, no other devices can use the bus; therefore, the talker function is turned on by the controller when necessary.

Any device on the bus is designated a "listener" when its primary function is the execution of commands. A printer is such a device. If a listener accepted all commands and data transmitted on the bus, chaos would result; therefore, the listener function must also be turned on by the controller. Some listeners also have a talker function.

The "controller" is the device that arbitrates all transfers of data on the bus. The controller is the device that can turn on a talker for the transmission of data. Likewise, the controller is the device that can turn on a listener for the reception of data.

Only one controller is allowed on this implementation of the bus. This controller must be your Apple.

A controller is able to interact with both talkers and listeners in various ways. The exact capabilities of the Apple as controller are detailed in the chapter on programming the GPIB.

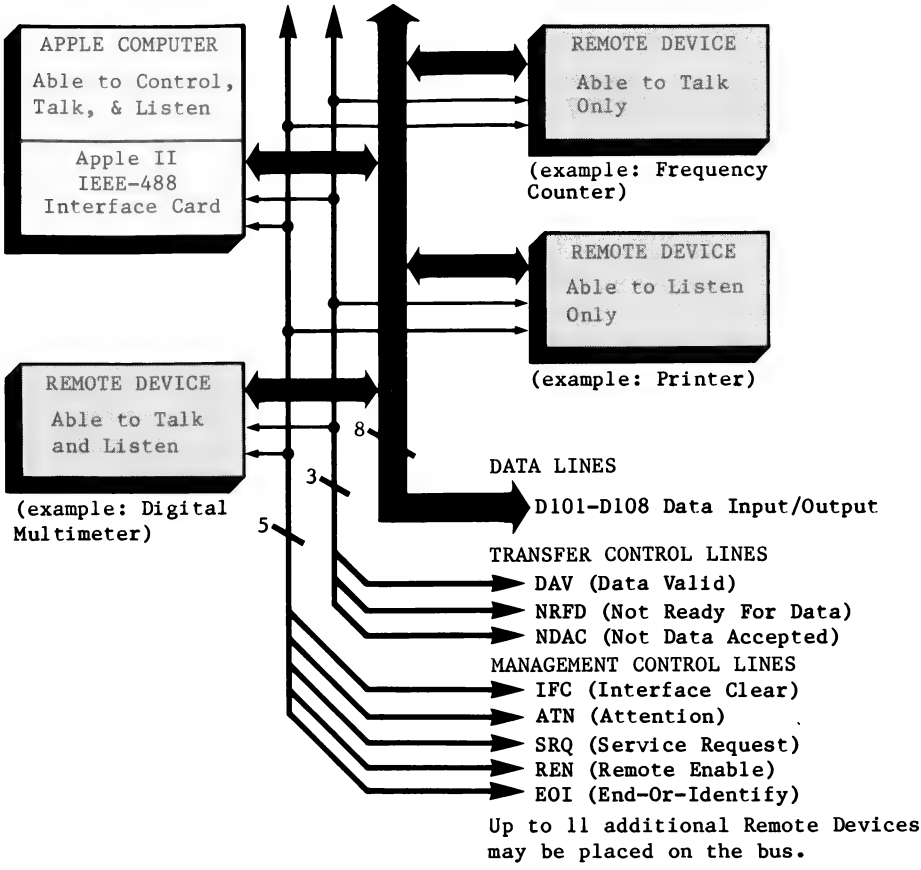


Figure 2-1. Apple II GPIB Functional Block Diagram.

PHYSICAL REQUIREMENTS

We have already stated that the Apple II GPIB can control up to 14 external devices. There are some limits on the total length of the bus and the distance between devices. The standard states that the total length of the bus may not exceed two meters times the number of devices on the bus. The maximum length of the entire bus is 20 meters.

The devices on the bus can be connected in either a daisy chain or a star configuration using the stackable connectors of the available standard bus cables. Daisy chain connections are preferred by most system designers, since the star connector stack may be unwieldy in larger systems. A sketch of star and daisy chain connected equipment are shown in Figures 2-2 and 2-3.

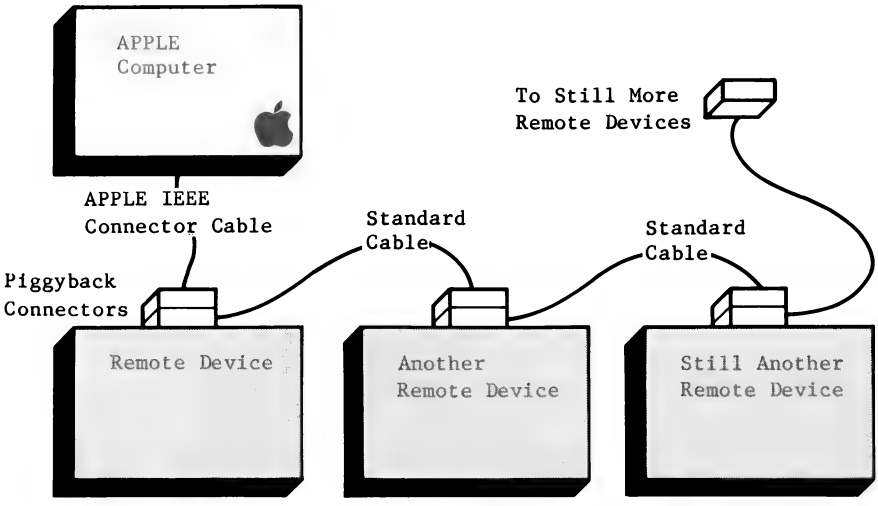


Figure 2-2. Daisy Chain Configuration.

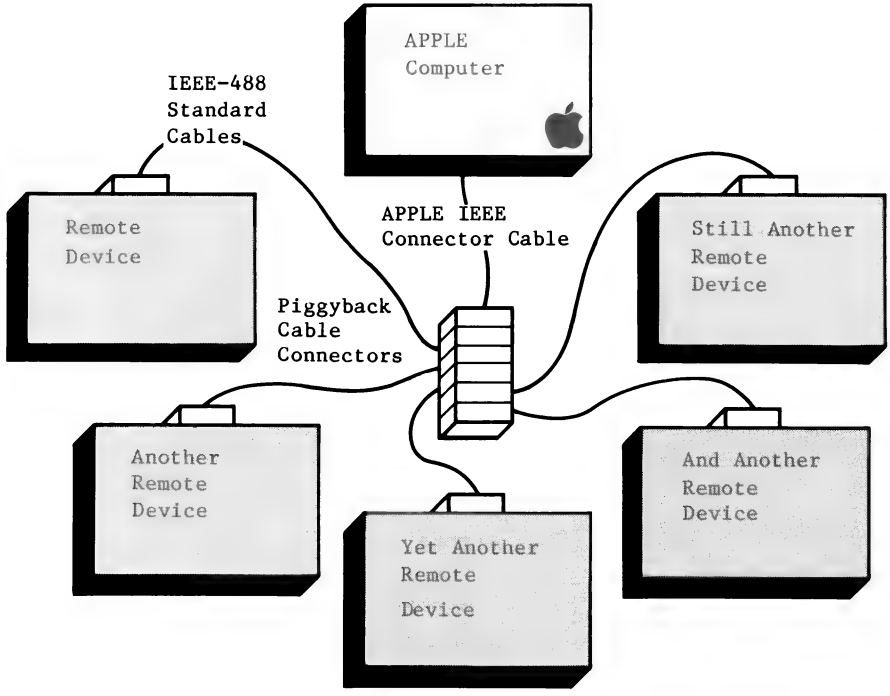
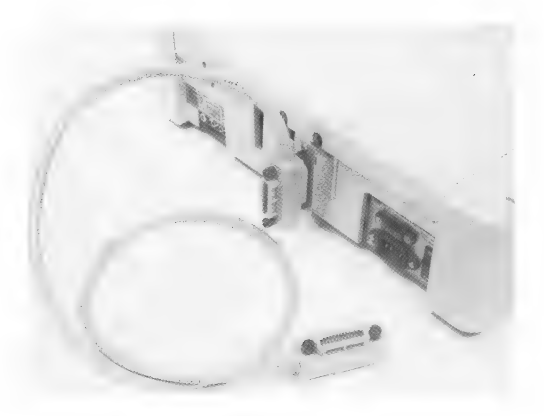


Figure 2-3. Star Connected Configuration.

Stackable or piggyback cables in various lengths are available from electronic supply houses and, in some cases, from your Apple dealer.

A typical commercial cable assembly is shown in the photograph.



DATA RATES

The GPIB is a parallel bus which transmits eight-bit bytes of data at high data rates. The GPIB can also transmit blocks of commands or data, one byte at a time. Block transmission allows rapid transfer of data with a minimum number of time-consuming handshake and control functions. In other words, control overhead is at a minimum. The average block size sent by a typical instrument is 13 bytes. A printer or disk drive is able to send or receive an entire line or sector in a single block.

The theoretical maximum rate of data transfer over the bus is one megabyte per second. The practical maximum is approximately 250 kilobytes per second during a Direct Memory Access (DMA) operation. Usually, a DMA requires so much time to set up that the speed advantage is lost for the short blocks of data usually transferred in GPIB operations.

The typical data rate for instruments on the bus is about one to two kilobytes per second. The operation of the GPIB is most efficient, and program overhead is minimized if all of the devices on the bus have buffered inputs; that is, if they can accept a block of data as fast as the controlling Apple can send it.

ELECTRICAL REQUIREMENTS

Ideally, devices on the bus that are not in use should be turned off to conserve energy and to increase the device's operating life. However, the best compromise between ideal energy conservation and proper operation of the bus requires that only some of the devices be turned off. The rule of thumb is that the Apple itself, plus at least half the devices connected to it, must have power on.

GPIB CONNECTIONS

The interface bus consists of eight data lines and eight control lines. The organization of these lines on the standard bus cable is shown in Table 2-1. The mnemonic symbols given for the connections are explained in the following paragraphs.

PIN	SIGNAL
1	DIO1
2	DIO2
3	DIO3
4	DIO4
5	EOI
6	DAV
7	NRFD
8	NDAC
9	IFC
10	SRQ
11	ATN
12	shield (earth ground)
13	DIO5
14	DIO6
15	DIO7
16	DIO8
17	REN
18	logic ground
19	logic ground
20	logic ground
21	logic ground
22	logic ground
23	logic ground
24	logic ground

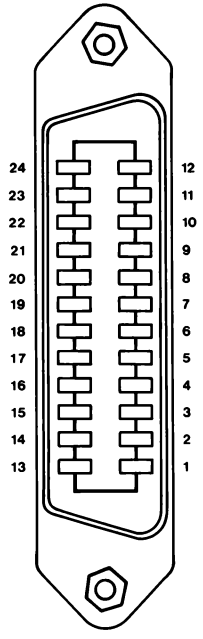


Table 2-1. Standard Bus Connector Pin Assignments.

DATA LINES

There are eight data lines in the GPIB. The data lines are numbered DIO1 (Data Input/Output 1) through DIO8 and correspond to the data bits Ø through 7 used by the Apple. The data carried by the data lines is represented using active low or negative logic. In negative logic, a low (Ø volt) level on a line corresponds to a True and a high (5 volt) level corresponds to a False. The interface card inverts the data before sending it to and after receiving it from the Apple; thus all the data the Apple sends and receives is positive true logic.

MANAGEMENT CONTROL LINES

The GPIB uses eight lines to control the devices on the bus and to direct the flow of data on the bus. Five of the control lines are used for the management of data transfers. The remaining three control lines are used for handshaking, or coordination of the transfer of individual bytes of data. A brief description of the five management control lines is given below. The mnemonic name for each signal is followed by its meaning.

The signals of all control lines are active low; that is, a line is considered to be active, or True, when its value is low (Ø volts).

For most applications you need not know which control lines are used with each command. The appendix on bus input/output sequences gives examples of the protocol used by each command.

ATN ATteNtion

The controller (Apple) indicates that an address or control byte is being placed on the bus by setting the ATN line true. The ATN line is set false to allow a device to place data on the bus. The ATN line is also used with the EOI line to initiate a parallel poll. The ATN line is set synchronously with the control handshake to eliminate confusion between control and data bytes.

EOI End Or Identify

The EOI signal has two uses. 1) The controller (Apple) sets EOI true and ATN true to initiate a parallel poll. 2) A talker will set EOI true with the last byte of data to indicate the end of the transmission. Unfortunately, not all available devices use the EOI to end a transmission.

SRQ Service ReQuest

The SRQ signal is set true by any device on the bus which requires some action from the controller. Upon reception of the SRQ signal, the controller software places the address of each device in turn (serial poll) on the data lines of the bus. Each device returns one status byte. When the device that set SRQ true detects its address on the bus, it identifies itself by returning a status byte with DIO7 true, and then it releases the SRQ line.

IFC InterFace Clear

The IFC signal is used by the controller (only) to set all device interfaces to a known initial state. This signal is used by the Abort command only.

REN Remote ENable

The REN signal is used by the controller (only) to allow a device to be remotely controlled. The device remains remotely controlled as long as the REN signal remains true. The device will not respond to data sent on the bus unless the REN signal is true.

TRANSFER CONTROL LINES

Data transfer on the bus is controlled by the three transfer control lines. The three lines use a three-wire handshake system (patented by Hewlett-Packard). The handshake system is asynchronous, allowing devices with both fast and slow data rates to operate on the same bus. The three-wire handshake is used to transfer every byte of information that is passed over the data lines. The three transfer control lines are described below. A diagram of the handshaking protocol is shown in figure 2-4.

NRFD Not Ready For Data

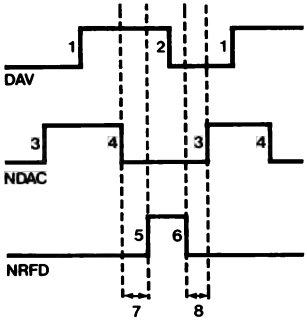
The NRFD signal is set true (low) by a listener to indicate that it is not yet ready for the next data or control byte. The NRFD signal at the controller will not be sensed as false (that is, ready for data) until all listeners on the bus are ready.

NDAC Not Data ACcepted

The NDAC signal is set true (low) by a listener to indicate that it has not yet accepted the data or control byte on the data bus. The NDAC signal at the controller will not be sensed as false (that is, data accepted) until all listeners on the bus have accepted the data.

DAV DAta Valid

The DAV signal is set true (low) by a talker to indicate that a data or control byte is on the data bus and is available to all listeners.



1. Data No Longer Valid -- Talker lets DAV go high when it sees that all listeners have accepted data (3). Talker may now change the data.
2. Data Valid -- Talker sees all listeners are ready for data (5). It pulls DAV low when the new data is ready.
3. Data Accepted -- The last listener has accepted the data and then released NDAC.
4. Acknowledge Data Not Valid -- The first listener that sees that data is valid (2) pulls NDAC low.
5. Ready For Data -- The last listener has seen that data is no longer valid (1) and has released NRFD to show that it is ready to accept data.
6. Not Ready For Data -- The first listener to accept data after data becomes valid (2) pulls NRFD low.
7. When it sees that data is not valid (1), each listener pulls NDAC low (4) and simultaneously releases NRFD (5).
8. When it sees that data is valid (2), each listener pulls NRFD low (6) and simultaneously releases NDAC.

Figure 2-4. Three-wire Handshaking Protocol.

GPIB FUNCTIONS

The Apple II GPIB has eleven general functions which control devices on the bus and transmit data. Every IEEE-488 compatible device is able to perform at least one of these functions. Please note that not all of the devices you may want to use on the bus will be able to use all of these functions. The general functions are briefly described in the following paragraphs.

SH1 Source Handshake

The source handshake function uses the three transfer lines to allow a talker (other than the controller) to transfer data to one or more listeners or the Apple on the bus.

AH1 Acceptor Handshake

The acceptor handshake function uses the three transfer lines to allow a listener to accept data from a talker which is usually the Apple.

T3 BASIC Talker

The controller uses the control and data lines to address a device on the bus and enable the device to be a talker. When a device is enabled as a talker, the device can send status and data bytes on the bus. This function name has no relationship to the BASIC language.

L1 BASIC Listener

The controller uses the control and data lines to address a device on the bus and enable the device to be a listener. When a device is enabled as a listener, the device can receive commands and data from the bus. This function name has no relationship to the BASIC language.

C4 Respond to Service Request

The service request function uses the SRQ control line to notify the controller (the Apple) of action required.

C3 Remote Enable

The remote enable function will enable a device to be controlled by the Apple rather than by the device front panel controls.

C25 Parallel Poll

The parallel poll function uses the eight data bus lines to poll up to eight devices on the bus simultaneously. A parallel poll is used to determine whether a device needs to be serviced by the Apple, and if so, which device. The parallel poll is much faster than the service request function, but requires that the Apple conduct frequent parallel polling operations. Many currently available devices do not have parallel poll capabilities.

DC Device Clear

The device clear function permits the controller to clear or initialize (set to a known state) a device on the bus. The device clear function is separate from the interface clear function, which is controlled by the IFC line.

C2 Interface Clear

The interface clear function is used to set the GPIB interface card controller circuit, and all devices, to a known state prior to use.

DT Device Trigger

The device trigger function is used by the controller to synchronize the operation of one or more devices on the bus. The Apple can initiate a group execute trigger (GET) to synchronize a number of devices on the bus. The group execute trigger is not to be confused with the GET instruction in BASIC.

C1 Controller

The controller function allows the controlling device (the Apple) to send addresses and universal and addressed commands to the devices on the bus. Only one system controller is allowed on the bus. The Apple is the system controller.

CHAPTER 3
PROGRAMMING THE GPIB

This chapter contains the programming concepts necessary to write a BASIC program for the GPIB. It begins with a section which explains a little about how the bus works, and how devices generally expect the bus to operate. Next is a summary of the addressing scheme used by the Apple II GPIB, followed by an in-depth description of each command that is available on the GPIB. The final section of the chapter discusses error messages and their causes.

GETTING STARTED

This section briefly explains how to get started programming the IEEE-488 card, some common features of different instruments, and what you should look for in the documentation of the instrument you are interfacing to the bus. It is a general introduction to the bus commands that are described later. All commands referred to in this section will be explained later.

Before you can use the bus, you must issue a PR#n or IN#n command (for the IEEE-488 card in slot n). Every time a PR#n or IN#n is used, the default settings of the card are set to the following values: screen off (SCØ), linefeed off (LFØ), device number Ø (DVØ), and end of string character equal to carriage return (<CR>). If, for example, you do a PR#Ø and IN#Ø to ask a question of the operator, and then wish to direct I/O back to the card without first resetting the default values, you may use the subroutine in Program 3 of Appendix D.

The format of the output data generally differs from one device to another. Some devices send out a <CR><LF> at the end of a string, while others send out only a <CR>. The appropriate linefeed command, LF1 or LFØ, must be used to ensure proper reception of the data. Some devices send out commas or colons as delimiters between strings of data, or between a function code and a data string. Applesoft terminates input of the current string when one of these characters is detected; therefore, for each comma or colon in the input string, the INPUT statement must have an additional variable. If the number of commas or colons sent by the device is variable, you may want to use the subroutine, Input With Commas, in Program 4 of Appendix D.

Some devices must be initialized with certain commands before they can be used for input. Some instruments require a <CR><LF> for setup, and others just need a <CR>. Some devices must be set into "remote" mode before they will respond to commands sent on the IEEE-488 bus, others require "local lockout" mode as well. The commands RA, RM, LA, LL and LO are used to set devices to these different modes.

Certain instruments must receive a trigger command (TG) before they will send a reading to the bus. The data is usually latched into the instrument at the time of the trigger, and held until it is read by some device on the bus. The read operation clears the latch until another trigger command is issued to that device. Some instruments might give false readings if they are not triggered before they are read.

Other commands are used to put the bus and the instruments on it into a known state. These commands are CA, CL, UT and AB. They will reset certain of the instrument's modes, usually those which determine whether the device is set up to receive or to send data.

These are all typical conditions which need to be attended to when a device is being used on the bus. In all cases the instrument's instruction manual will explain exactly which commands the device must receive in order to operate properly.

Because this section deals with commands that you know very little about, we suggest that you re-read it after you have studied the command descriptions later in this chapter.

COMMAND NOTATION

The standard form of an GPIB command is a string of characters; each character is a 7 bit ASCII code as sent by the Apple keyboard. Each character is sent or received in the low seven bits of one byte. The general format of a command string is

<command> <control characters> <data>

The <command> is a two letter mnemonic device used to execute one of the GPIB commands. The <control characters> and <data> parameters are required by some of the commands, and not required by others.

The commands used by the GPIB are summarized in Table 3-1. The left column lists the command names; the column labelled FORMAT lists the form of each command string in Backus-Naur Form (BNF); and the column labelled BASIC lists the BASIC command (PRINT or INPUT) that must be used to send the command.

All GPIB commands are sent by BASIC to the IEEE-488 card. Depending on the command sent, the card may or may not transfer data on the bus. The rightmost column of Table 3-1, labelled BUS?, says YES if the command in that row of the table causes data to be placed on or read from the bus.

BNF descriptions of the <control characters> and <data> portions of the command strings are given in Table 3-2. Every bracketed component, called a descriptor, is defined in Table 3-2 in terms of ASCII characters and other descriptors.

In BNF notation, a symbol that is not enclosed in brackets should be included in the command string literally. A descriptor enclosed in brackets is used to represent an element, or list of elements, that must occupy that position in the command string. Descriptors that are separated by a vertical bar (|) are interchangeable. A vertical bar may be read as the English word "or".

For clarity, elements of the command string are separated by spaces, both in the tables and in the individual command descriptions later in the chapter. Actual command strings must not contain these spaces since a space can be interpreted as the address of a listener.

In Table 3-2, note especially that the <listen list> and <count> parameters must be terminated by the character <CTRL-Z>. This may not be obvious from Table 3-1. Note also the use of the symbols <CTRL-Z>, <CR>, and <SP>. These symbols refer to pressing Z while holding down the key labelled CTRL, pressing the key labelled RETURN, and pressing the spacebar, respectively.

Tables 3-1 and 3-2 are also included in the reference card in the back of the manual.

Table 3-1. GPIB Command Summary.

COMMAND	FORMAT	BASIC	BUS?
WRITE	WT <listen list> <data> <EOS>	PRINT	YES
WRITECNT	WC <listen list> <count> <data>	PRINT	YES
READ	RD <talker> <CTRL-Z> <data> <term>	INPUT	YES
READCNT	RC <talker> <CTRL-Z> <count> <data> <term>	INPUT	YES
XFER	XF <talker> <CTRL-Z> <listen list> <CR>	PRINT	YES
TRIGR	TG <listen list> <CR>	PRINT	YES
CLRAL	CA <CR>	PRINT	YES
CLEAR	CL <listen list> <CR>	PRINT	YES
REMAL	RA <CR>	PRINT	YES
REMDV	RM <listen list> <CR>	PRINT	YES
LLKAL	LL <CR>	PRINT	YES
LOCAL	LA <CR>	PRINT	YES
LOCDV	LO <listen list> <CR>	PRINT	YES
SRQD	SR <srq> <CR>	INPUT	YES
SPOLL	SP <talker> <CTRL-Z> <status word> <CR>	INPUT	YES
PPOLL	PP <status word> <CR>	INPUT	YES
PPENB	PE <enable list> <CR>	PRINT	YES
PPDIS	PD <listen list> <CR>	PRINT	YES
PPUAL	PU <CR>	PRINT	YES
ABORT	AB <CR>	PRINT	YES
LINEFEED	LF <off/on> <CR>	PRINT	NO
EOS	ES <EOS> <CR>	PRINT	NO
SCREEN	SC <off/on> <CR>	PRINT	NO
DEVICE	DV <device number>	PRINT	NO
UNTALK	UT <CR>	PRINT	YES

Table 3-2. GPIB Command String Definitions.

<listen list> ::= <CTRL-Z> | <listener> <CTRL-Z> | <listener> <listen list>

<listener> ::= <primary listener> | <primary listener> <secondary address>

* <primary listener> ::= <SP> | ! | " | # ... > | ?

* <secondary address> ::= ' | a | b | c ... } | ~

<count> ::= <decimal digit> <CTRL-Z> | <decimal digit> <count>
NOTE: The maximum total value for count is 255.

<EOS> ::= any ASCII character

<data> ::= empty | any ASCII character | any ASCII character <data>

<talker> ::= empty | <primary talker> | <primary talker> <secondary address>

* <primary talker> ::= @ | A | B | C ... ^ | _

* <hex digit> ::= Ø | 1 | 2 ... 9 | A | B ... F

* <decimal digit> ::= Ø | 1 | 2 ... 9

<status word> ::= <hex digit> <hex digit>

<srq> ::= ASCII "T" | ASCII "F"

<term> ::= empty | <CR> | <EOS> | EOI
NOTE: empty if last char = <CR> or <CR><LF>

<device number> ::= <decimal digit> <decimal digit>
NOTE: Maximum device number is 3Ø

<off/on> ::= ASCII "Ø" | ASCII "1"

<enable list> ::= <listener> <enable> | <listener> <enable> <enable list>

* <enable> ::= @ | A | B ... N | O

<CR> ::= ASCII carriage return (13 decimal)

<LF> ::= ASCII linefeed (1Ø decimal)

<SP> ::= ASCII space (32 decimal)

<CTRL-Z> ::= ASCII SUB (26 decimal)

<CTRL-D> ::= ASCII ET (4 decimal)

* NOTE: Marked elements are ASCII characters.

ADDRESSES

There are three types of addresses which may be included as part of the <control characters> parameter. These address types are: listener address, talker address and secondary address. Each address is specified by a distinct character corresponding to one of the ASCII codes. Address characters referenced in the following text are enclosed in quotation marks.

LISTENER ADDRESSES

The addresses of all listeners on the GPIB are restricted to the hex values 2Ø through 3E. The listener address range corresponds to ASCII characters (as sent from the keyboard) from <SP> through ">". The decimal, octal, and hexadecimal values of each possible listener address are shown in Table A-1 of Appendix A. Hex value 3F (ASCII "?") is reserved as a special universal UNListen (UNL) command which is automatically sent by the Apple II GPIB when needed.

TALKER ADDRESSES

There may only be a single talker at a time. The address of the talker on the GPIB is restricted to the hex values 4Ø to 5E. The talker address range corresponds to ASCII characters (as sent from the keyboard) from "@" through "^". The decimal, octal, and hexadecimal values of each possible talker address are given in Table A-2 of Appendix A. Hex value 5F (ASCII "_") is reserved as a special universal UnTalk command which is sent with the UT command.

SECONDARY ADDRESSES

Secondary addresses may be used to set operating features of some devices. They may also be used to address subunits of a complex device. For instance, the operating range and measurement type for some remote digital voltmeters may be set from the Apple by means of secondary addresses.

The GPIB provides the capability of sending a secondary address to either talkers or listeners. Secondary addresses are included as elements of the <talker> or <listen list> parameters if they are needed. A secondary address must immediately follow the primary address with which it is associated.

Secondary addresses on the GPIB are restricted to the hex values 6Ø through 7E. The secondary address range corresponds to ASCII characters (as sent from the keyboard) from "`" through "~". The decimal, octal, and hexadecimal values of each possible secondary address are shown in Table A-3 of Appendix A.

From APPLESOFT, the characters needed for secondary addresses are not directly available from the keyboard. They can, however, be generated using the CHR$ function. The proper CHR$ calls are shown in Table A-3 of Appendix A. Examples of the use of this function are given in the examples of the commands.

From Integer BASIC, a special subroutine must be used to generate secondary addresses. This subroutine is listed in Program 1 of Appendix D.

ON USING ADDRESSES

You must carefully read the operating manual of the device you are planning to use with the GPIB, to ensure that you understand the addressing method used by that device. The device's addresses might be preset by factory-installed jumpers, or they might be adjustable using addressing switches. It is important that you determine what address values the manufacturer intended the device to have.

The address for the device may be given in binary, hex, octal, or ASCII. You must determine what character, when sent from the Apple keyboard, will represent that address. If the address is not hard-wired, you should set the address switches or jumpers to a convenient address. Tables of addresses are given in Appendix A.

Each device has associated with it a device number assigned by the manufacturer or set as explained above. The device number is only used internally to the IEEE-488 card.

Be sure that no instrument has the same device number as the Apple. The default device number for the Apple is Ø. Device number Ø (not ASCII Ø) corresponds to a listener address <SP> and a talker address "@".

GPIB COMMAND DESCRIPTIONS

Table 3-1 lists all the commands that you can send to the Apple II GPIB. There are three types of commands: output commands, input commands, and commands used to set operating characteristics of the IEEE-488 card. The last type does not cause any data to be placed on the bus; the others do. The two rightmost columns of Table 3-1 indicate which type each command is. The column labelled BASIC tells which BASIC command, PRINT or INPUT, should be used to send the command in each row.

SENDING A COMMAND

Commands which send data to a device take the form

```
PRINT "<command><control chars><data>"
```

and commands which read data from a device take the form

```
INPUT "<command><control chars>"; S$
```

or

```
PRINT "<command><control chars>";
INPUT S$
```

in which S$ is a string appropriate for that particular command.

A program that sends commands to the GPIB must first indicate the number of the slot into which the interface card is inserted. In the examples which follow, the slot number is assumed to be 3. The statement

```
PR#3
```

must precede the first statement in which you PRINT a command to the card. Likewise the statement

```
IN#3
```

must precede the first statement in which you INPUT data from the card. Quite simply, you are telling the Apple to talk to or listen to the Apple II GPIB, which is in slot 3 (in this case).

A DOS command from a program must appear in a PRINT statement whose first output character is <CTRL-D> and whose output is separated from preceding and from succeeding printed output by <CR>s.

Thus, if you are using DOS, the statements

```
1Ø    D$ = "": REM  A <CTRL-D> LIES BETWEEN THE QUOTES
2Ø    PRINT D$;"PR#3"
3Ø    PRINT D$;"IN#3"
```

must precede the first PRINT and INPUT statements to the card. The DOS form of the commands is the one that we will use in the following examples.

The commands PR#n and IN#n have the effect of resetting all the card's default values. The resetting of default values can be avoided with the subroutine in Program 3 of Appendix D.

COMMAND PROTOCOL

As previously noted, <listen list> and <count> must be terminated by the character <CTRL-Z>. For the following examples, assume that the variable Z$ has been set to the value <CTRL-Z> using one of the BASIC statements

```
5    Z$ = ""         :  REM  <CTRL-Z> IS BETWEEN THE QUOTES
5    Z$ = CHR$(26)   :  REM  CHR$(26) IS <CTRL-Z> IN APPLESOFT
```

and that D$ has been set to the value <CTRL-D> as shown above.

When using the GPIB with the SCREEN option set to ON, a <CTRL-Z> prints out as an inverse-video Z. See the SCREEN command later in this chapter for more details.

A command string is terminated by a <CR> or an end-of-string character, <EOS>. The PRINT command normally sends a <CR> as its last character. If <EOS> is not <CR>, or if you want to PRINT a string containing characters as well as variables, use the special symbol ";" to continue the PRINT statement without sending a <CR>. For example, to print the character sequence "WT1" followed by <CTRL-Z> and then A$, without <CR>s between them, use

```
10   PRINT "WT1" ; Z$ ; A$ :  REM THE VALUE OF Z$ IS <CTRL-Z>
```

This is equivalent to the statements

```
10   PRINT "WT1";
11   PRINT Z$;
12   PRINT A$
```

It is, of course, possible to embed control characters into any string, but since embedded control characters are not visible, the above convention was adopted.

BUS TRANSFERS

When a command is executed by the GPIB, a sequence of data and control signals is transferred on the 16 data and control lines. Examples of the input/output sequences for all the GPIB commands are given in Appendix B. The bytes sent on the DIO lines are data and IEEE-488 commands. Refer to the IEEE standard for more details.

GPIB COMMANDS

The first four commands given below -- LINEFEED, EOS, SCREEN and DEVICE -- are commands that set operating characteristics of the card. Although these commands do not cause any data to be sent on the bus, some of them do have an effect on data that is sent over the bus by other commands.

The rest of the commands cause some sort of data to be transferred over the bus: this data can take the form of information passed from one device to another, or it can take the form of commands that set operating characteristics of the devices on the bus.

LINEFEED: LF

Linefeed Control. The LINEFEED command is used to set the board to a mode that will generate a line feed character (<LF>) after each carriage return (<CR>). The LINEFEED command takes the form

```
LF <off = 0/on = 1> <CR>
```

During WRITE commands issued while LINEFEED is set to on, the IEEE-488 card will send a <LF> following every instance of <CR> in

<data>, even if <data> already has a <LF> after each <CR>. The LINEFEED option does not affect the data sent by WRITECNT commands. During READ commands issued while LINEFEED is set to on, the IEEE-488 card expects a <LF> to follow the terminating <CR>.

The LINEFEED commands are as follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "LF1"  (line feed character is provided after each <CR>)

2Ø   PRINT "LFØ"  (line feed character is not provided after each <CR>)
```

LINEFEED is set to off when <EOS> is changed from <CR>.

EOS: ES

End Of String Character. The EOS command is used to change the character used to terminate a string. The EOS character is a <CR> under normal or default conditions. A reset or PR#3 instruction resets the EOS character to <CR> . LINEFEED is set to off when <EOS> is changed from <CR>. The EOS command is

ES <EOS> <CR>

The EOS command format is as follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "ES!"
```

This command changes the EOS character from <CR> to !.

SCREEN: SC

Screen Control. The SCREEN command is used to control the CRT display during operation of the GPIB. The CRT display shows all input and output characters when SCREEN control is on. The display is very useful during program debugging. The normal (default) condition of SCREEN is off. The SCREEN command takes the form

SC <off = Ø/on = 1> <CR>

The SCREEN command is as follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "SC1"    (data is displayed on the CRT display)

2Ø   PRINT "SCØ"    (data is not displayed on the CRT)
```

DEVICE: DV

Controller Device Number. The DEVICE command is used to set the device number of the Apple controller. The normal or default device number of the Apple controller is Ø, and it will be reset to Ø by a PR#n instruction, or by pressing RESET. The device number, used modulo 32, may be one or two ASCII decimal digits; if the number is Ø9 or less, the leading Ø can be omitted. The device number 31 (corresponding to 31,63,95...) may not be used because it is reserved for the UNLISTEN and UNTALK commands. The DEVICE command takes the form

```
DV <device number> <CR>
```

The DEVICE command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "DVØ1"
```

WRITE: WT

Write Data Out. The WRITE command sends data to all devices named in <listen list>. The WRITE command is the normal way of sending data from the Apple to one or more remote devices. The command takes the form

```
WT <listen list> <data> <EOS>
```

An EOI (End-Or-Identify) bus signal is automatically sent with the last <EOS> character. A <CR> is the normal <EOS>. If the LF flag is set (see the LF command), and the <EOS> character is <CR>, then EOI is sent with the <LF>. Sample WRITE commands follow:

```
1Ø   D$;"PR#3": REM  THE VALUE OF D$ IS <CTRL-D>
3Ø   PRINT "WT1"; Z$; A$ : REM  THE VALUE OF Z$ IS <CTRL-Z>
```

This command sends the string of data, A$, to the listener with address equal to the value of keyboard character 1 (49 decimal, 61 octal or 31 hex).

```
4Ø   PRINT "WT8"; CHR$(97); Z$; A$ : REM  THE VALUE OF Z$ IS <CTRL-Z>
```

This command sends the string of data, A$, to the listener with primary address "8" and secondary address "a". The CHR$ function must be used to send lower-case characters such as "a" from the Apple keyboard. The CHR$ functions are listed in table A-3 of Appendix A. The ASCII code for "a" is 97, so CHR$(97) generates an "a".

Program 1 in Appendix D shows how lower case characters can be sent from Integer BASIC.

WRITECNT: WC

Write Data Out With Count. The WRITECNT command is like a WRITE command that includes a count of the number of bytes in the data string. The command takes the form

```
WC <listen list> <count> <data>
```

The <EOS> character (nominally <CR>) is not normally sent at the end of the data string, but the bus signal EOI is sent with the last character. The count must be greater than Ø and less than 256. A sample WRITECNT statement follows:

```
1Ø   D$;"PR#3": REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "WC1"; Z$; LEN(B$); Z$; B$; :REM THE VALUE OF Z$ IS <CTRL-Z>
```

This command sends the string B$ of length LEN(B$) to the listener with address "1".

One important use of the WRITECOUNT comand is to send data that is not terminated by the <EOS> character. This cannot be done with the WRITE command.

Any extra characters beyond the count will be interpreted as a new command and will cause an error. Too few characters will cause the next command to be sent as part of the data string and will cause an error. If the PRINT statement is not terminated by a ";", a <CR> will be automatically sent and must be included in the character count. Likewise, if LINEFEED is set, <CR><LF> must be included in the count.

READ: RD

Read Data In. The READ command receives data from a talker specified by the command. The READ command is the normal method of receiving data from a remote device. The command takes the form

```
RD <talker> Z$ <data> <term>
```

The input data string terminates when a <CR> or <EOS> character (nominally a <CR>) is received, or when the EOI signal goes true. The IEEE-488 card will always return a <CR> to the program to terminate the input data. If the last character from the card is a <CR>, an additional <CR> will not be sent. A line feed character from the GPIB will be suppressed and not passed to the program. A sample READ command follows:

```
5    D$;"PR#3" :REM  THE VALUE OF D$ IS <CTRL-D>
1Ø   D$;"IN#3" :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "RDA"; Z$;  : REM  THE VALUE OF Z$ IS <CTRL-Z>
25   INPUT C$
```

This command receives the data string C$ from the talker with address "A".

```
5    D$;"PR#3" :REM  THE VALUE OF D$ IS <CTRL-D>
1Ø   D$;"IN#3" :REM  THE VALUE OF D$ IS <CTRL-D>
3Ø   PRINT "RDA"; CHR$(98); Z$;  : REM  THE VALUE OF Z$ IS <CTRL-Z>
35   INPUT C$
```

This command receives a data string C$ from the talker with address "A" and secondary address "b". The lower case "b" is generated by CHR$(98).

Note the use of the ";" following the last two PRINT statements. If a PRINT statement sends a command and an INPUT statement reads the data placed on the bus because of that command, it is necessary to suppress the <CR> normally sent after the PRINT.

READCNT: RC

Read Data In With Count. The READCNT command is like the READ command with a field, count, that specifies the length of the input string in bytes. The command takes the form

RC <talker> Z$ <count> <data> <term>

EOI may be sent with the last character in the string by the talker. The data string may contain less than count characters if the talker uses EOI or <EOS>.

The IEEE-488 card will always return a <CR> to the program to terminate the input data. If the last character from the card is a <CR>, an additional <CR> will not be sent. An example of a READCNT command follows:

```
 9   D$;"PR#3" :REM  THE VALUE OF D$ IS <CTRL-D>
1Ø   D$;"IN#3" :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "RCB"; Z$; "26"; Z$;  : REM  THE VALUE OF Z$ IS <CTRL-Z>
25   INPUT C$
```

This command receives the data string C$, containing 26 characters, from the talker with address "B".

If you are using the RC or RD command with the LINEFEED option set (see the LF command), be sure that the device does not send a NULL (ASCII ØØ) when the GPIB is expecting a <LF>.

XFER: XF

Transfer Data. The XFER command is used to transfer data between remote devices other than the Apple. The talker is addressed first, followed by the listeners. The device address of the Apple (nominally Ø) must NOT appear as the talker or in the listen list. The XFER command takes the form

```
XF <talker> Z$ <listen list> <CR>
```

The talker MUST be programmed to send EOI with the last character in the transfer. This tells the listener(s) that the transfer is complete, and allows the Apple to regain control of the GPIB. A sample XFER command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "XFA"; Z$; "1"; Z$
```

This command allows the talker with address "A" to transfer data directly to the listener with address "1".

TRIGR: TG

Group Execute Trigger. The TRIGR command is used to configure or set up a group of remote devices on the GPIB and to cause all of the devices to commence execution at the same time. The command takes the form

```
TG <listen list> <CR>
```

A sample TRIGR command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "TG"; "19"; Z$
```

This command causes listeners with addresses "1" and "9" to be triggered simultaneously.

CLRAL: CA

Clear All Devices. The CLRAL command issues a universal clear message to all remote devices on the GPIB. The command format is

```
CA <CR>
```

An example of the CLRAL command is

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "CA"
```

This command clears all devices on the GPIB.

CLEAR: CL

Clear Selected Devices. The CLEAR command clears the remote devices specified by the address list. The command takes the form

```
CL <listen list> <CR>
```

A sample CLEAR command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "CL"; "19"; Z$
```

This command issues a clear message to the remote devices with addresses "1" and "9".

REMAL: RA

Remote Enable All. The REMAL command is used to set the REN (remote enable) control line true. The remote devices will not be in remote operating mode until addressed to listen. The REMAL command is

```
RA <CR>
```

An example of a REMAL command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "RA"
```

All subsequent listen addresses will be sent with the REN signal true. The REN signal is not set to false until the LA command is sent.

REMDV: RM

Remote Enable Selected Devices. The REMDV command is used to set the REN (remote enable) line true and to send the address of the specified listeners on the GPIB. The listeners are placed in remote mode when addressed. The command takes the form

```
RM <listen list> <CR>
```

A sample REMDV command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
15   E$ = "12349" + Z$
2Ø   PRINT "RM"; E$
```

This command turns on remote operation of the devices with addresses "1", "2", "3", "4" and "9".

All subsequent listen addresses will be sent with the REN signal true. The REN signal is not set to false until the LA command is sent.

LLKAL: LL

Local Lockout All Devices. The LLKAL command is used to set all remoted devices on the GPIB to local lockout mode. The front panel controls of a remote device have no effect (e.g. if someone presses the instrument's RESET button) when the device is in local lockout mode. The LLKAL command follows:

```
LL <CR>
```

Here is an example of the LLKAL command

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "LL"
```

LOCAL: LA

Local Mode All Devices. The LOCAL command is used to set all remote devices on the GPIB to local operating mode. The command byte takes the form

```
LA <CR>
```

The LOCAL command will set the REN control line false. A LOCAL command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "LA"
```

LOCDV: LO

Local Mode Selected Devices. The LOCDV command is used to set selected remote devices to local operating mode. The LOCDV command takes the form

```
LO <listen list> <CR>
```

A sample LOCDV command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "LO"; "12349"; Z$
```

This command sets local operating mode for the remote devices with addresses "1", "2", "3", "4" and "9".

SRQD: SR

Service Requested. The SRQD command is used to determine whether any remote device on the GPIB is requesting service. The command will return the value of ASCII T (84 decimal, 124 octal, 54 hex) if the SRQ control line is true. The command will return the value of ASCII F (7Ø decimal, 1Ø6 octal, 46 hex) if the SRQ control line is false. The SRQD command takes the form

SR <srq> <CR>

The SRQD command follows:

```
5    PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
1Ø   PRINT D$;"IN#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   INPUT "SR"; G$
```

The value returned in G$ is either a "T" or a "F".

SPOLL: SP

Serial Poll. The SPOLL command is used to send a serial poll to an addressed remote device on the GPIB. The SPOLL is usually used when a "T" has been returned as the result of an SRQD command. The addressed remote device will return a status byte which will determine if that device has requested service. The IEEE-488 card translates the status byte into a 2 byte status word. The form of the SPOLL command is

SP <talker> Z$ <status word> <CR>

A sample SPOLL command follows:

```
5    PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
1Ø   PRINT D$;"IN#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "SPA"; Z$;
3Ø   INPUT H$
```

The value returned in H$ is a status word from the remote device with address "A". H$ is two bytes long. The first byte is the ASCII form of the most significant four bits of the binary status byte, representing a hexadecimal number between Ø and F. The second byte is the ASCII form of the least significant four bits of the binary status byte. <CTRL-Z> will not be displayed on the screen during SPOLL command. The value of the status byte returned by a device is determined by the manufacturer of that device.

If the addressed device was asserting SRQ, it will stop doing so and it will return the status byte with DIO7 asserted. This implies that the first character of H$ will be a 4, 5, 6, 7, C, D, E, or F. The meaning of other DIO bits is device dependent.

PPOLL: PP

Parallel Poll. The PPOLL command is used to conduct a parallel poll of remote devices. The result of the parallel poll is to return an eight bit status byte to the IEEE card. Each of the eight bits may be assigned to a device using the PPENB command for some devices, or by a DIP switch on devices that implement the PP2 subset of the IEEE-488 standard. The IEEE-488 card translates the status byte into a 2 byte status word. The form of the PPOLL command is

PP <status word> <CR>

A sample PPOLL command follows:

```
5    PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
1Ø   PRINT D$;"IN#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   INPUT "PP"; I$
```

The value returned in I$ is a two byte status word. The first byte is the ASCII form of the most significant four bits of the binary status byte, representing a hexadecimal number between Ø and F. The second byte is the ASCII form of the least significant four bits of the binary status byte.

PPENB: PE

Parallel Poll Enable. The PPENB command is used to set the configuration of remote devices which can implement the PP1 subset of the IEEE-488 standard. The command is used to assign a bit number and bit sense for each remote device to be enabled.

PE <enable list> <CR>

An element of the enable list consists of a listener address followed by an enable byte. The enable byte takes the form Ø 1 Ø Ø S $P_3 P_2 P_1$. Bit "S" of the enable byte assigns the sense of that listener's poll bit in the status byte (see above); that is, whether the device will signal a request for service with a 1 or a Ø. Bits $P_3 P_2 P_1$, treated as a number from Ø through 7, determine which bit of the status byte is assigned to that device. The enable byte corresponds to an ASCII character in the range "@" to "O". To conform with the IEEE standard, the IEEE-488 card ORs the status byte with a hexadecimal 2Ø before sending the byte on the bus. A sample PPENB command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "PE"; "1A2B3K4L"
```

The sample command results in the following configuration

Listener "1" signals requests with status bit 1, sense = Ø
Listener "2" signals requests with status bit 2, sense = Ø
Listener "3" signals requests with status bit 3, sense = 1
Listener "4" signals requests with status bit 4, sense = 1

PPDIS: PD

Parallel Poll Disable. The parallel poll disable is used to prevent the listed remote devices from responding to parallel poll commands. The PPDIS command takes the form

```
PD <listen list> <CR>
```

A sample PPDIS command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "PD"; "1234"; Z$
```

This command prevents the listeners with addresses "1", "2", "3" and "4" from responding to parallel poll commands.

PPUAL: PU

Parallel Poll Unconfigure All Devices. The PPUAL command is used to change the configuration of remote devices in order to prevent future responses to parallel polls. The PPUAL command is

```
PU <CR>
```

The PPUAL command is shown:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "PU"
```

ABORT: AB

Clear All Interfaces. The ABORT command is used only in cases of emergency to clear all device interfaces. It resets all devices to a known state by setting the IFC control line to true. The IFC pulse lasts for about 5 milliseconds. The ABORT command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "AB"
```

UNTALK: UT

Untalk. The UNTALK is used to send the Universal Untalk (UNT) command. This command is required for some remote devices. Carefully read the device's operating manual for actual requirements.

The UNTALK command follows:

```
1Ø   PRINT D$;"PR#3"  :REM  THE VALUE OF D$ IS <CTRL-D>
2Ø   PRINT "UT"
```

GPIB ERROR MESSAGES

When the GPIB encounters a stream of characters which it cannot understand, it issues an error message to the console. Execution of the program is terminated unless the BUS error occurs when the ONERR flag is set. There are four different error messages which are listed below along with typical causes of that particular error.

When displayed on your screen, the three letter error type is followed by the letters "ERR".

CMD — The first two characters of the command were not recognized as one of the GPIB commands. This could happen accidentally if, for example, <EOS> is no longer <CR> and a previous WRITE command didn't use a semicolon to suppress the final <CR>. It could also be caused by an unexpected comma, causing termination of input, in an input string. Another cause of the CMD error is the occurrence of a system error during an input command. The system error message starts with a "?" which is interpreted as an invalid command character.

LST -- The control characters and/or data do not follow the protocol required by the command being used. This error is typically caused by an omitted <CTRL-Z> or consecutive secondary addresses.

CNT -- The count is out of range (usually $\geq$ 256). This error can only occur with the WC, RC and DV commands.

BUS -- Incomplete source handshake error. It occurs if a byte is placed on the GPIB when NDAC and NRFD are both high. This situation will occur when the bus is improperly connected or if the addressed device is off. A BUS error can be recovered using the ONERR...GOTO statement.

THE ONERR . . . GOTO STATEMENT

Of the four error conditions that the GPIB can detect, only the BUS error is recoverable. If you are using APPLESOFT and your program includes an

```
ONERR GOTO linenum
```

statement, the GPIB firmware will detect the ONERR flag, and then transfer control to the proper place in the APPLESOFT firmware. APPLESOFT will then transfer control to linenum, as if a normal APPLESOFT error had occurred. You may insert any type of error recovery routine at the line indicated by linenum. A sample error routine is listed in Program 2 of Appendix D.

See the APPLESOFT II BASIC Programming Reference Manual for more details on the ONERR...GOTO statement.

APPENDIX A
ASCII CONVERSION TABLES

DEC = ASCII decimal code
OCT = ASCII octal code
HEX = ASCII hexadecimal code
CHR$ = Applesoft function

TABLE A-1 GPIB LISTENER ADDRESSES

DEC	OCT	HEX	CHAR	WHAT TO TYPE
32	Ø4Ø	2Ø	SPACE	space
33	Ø41	21	!	!
34	Ø42	22	"	"
35	Ø43	23	#	#
36	Ø44	24	$	$
37	Ø45	25	%	%
38	Ø46	26	&	&
39	Ø47	27	'	'
4Ø	Ø5Ø	28	(	(
41	Ø51	29	)	)
42	Ø52	2A	*	*
43	Ø53	2B	+	+
44	Ø54	2C	,	,
45	Ø55	2D	-	-
46	Ø56	2E	.	.
47	Ø57	2F	/	/
48	Ø6Ø	3Ø	Ø	Ø
49	Ø61	31	1	1
5Ø	Ø62	32	2	2
51	Ø63	33	3	3
52	Ø64	34	4	4
53	Ø65	35	5	5
54	Ø66	36	6	6
55	Ø67	37	7	7
56	Ø7Ø	38	8	8
57	Ø71	39	9	9
58	Ø72	3A	:	:
59	Ø73	3B	;	;
6Ø	Ø74	3C	<	<
61	Ø75	3D	=	=
62	Ø76	3E	>	>
63	Ø77	3F	?	Reserved for UNLISTEN

TABLE A-2 GPIB TALKER ADDRESSES

DEC	OCT	HEX	CHAR	WHAT TO TYPE
64	1ØØ	4Ø	@	@
65	1Ø1	41	A	A
66	1Ø2	42	B	B
67	1Ø3	43	C	C
68	1Ø4	44	D	D
69	1Ø5	45	E	E
7Ø	1Ø6	46	F	F
71	1Ø7	47	G	G
72	11Ø	48	H	H
73	111	49	I	I
74	112	4A	J	J
75	113	4B	K	K
76	114	4C	L	L
77	115	4D	M	M
78	116	4E	N	N
79	117	4F	O	O
8Ø	12Ø	5Ø	P	P
81	121	51	Q	Q
82	122	52	R	R
83	123	53	S	S
84	124	54	T	T
85	125	55	U	U
86	126	56	V	V
87	127	57	W	W
88	13Ø	58	X	X
89	131	59	Y	Y
9Ø	132	5A	Z	Z
91	133	5B	[	CHR$(91)
92	134	5C	\	CHR$(92)
93	135	5D	]	] (shift-M)
94	136	5E	^	^
95	137	5F	_	Reserved for UNTALK

TABLE A-3 GPIB SECONDARY ADDRESSES

DEC	OCT	HEX	CHAR	WHAT TO TYPE
96	14Ø	6Ø	'	CHR$(96)
97	141	61	a	CHR$(97)
98	142	62	b	CHR$(98)
99	143	63	c	CHR$(99)
1ØØ	144	64	d	CHR$(1ØØ)
1Ø1	145	65	e	CHR$(1Ø1)
1Ø2	146	66	f	CHR$(1Ø2)
1Ø3	147	67	g	CHR$(1Ø3)
1Ø4	15Ø	68	h	CHR$(1Ø4)
1Ø5	151	69	i	CHR$(1Ø5)
1Ø6	152	6A	j	CHR$(1Ø6)
1Ø7	153	6B	k	CHR$(1Ø7)
1Ø8	154	6C	l	CHR$(1Ø8)
1Ø9	155	6D	m	CHR$(1Ø9)
11Ø	156	6E	n	CHR$(11Ø)
111	157	6F	o	CHR$(111)
112	16Ø	7Ø	p	CHR$(112)
113	161	71	q	CHR$(113)
114	162	72	r	CHR$(114)
115	163	73	s	CHR$(115)
116	164	74	t	CHR$(116)
117	165	75	u	CHR$(117)
118	166	76	v	CHR$(118)
119	167	77	w	CHR$(119)
12Ø	17Ø	78	x	CHR$(12Ø)
121	171	79	y	CHR$(121)
122	172	7A	z	CHR$(122)
123	173	7B	{	CHR$(123)
124	174	7C	|	CHR$(124)
125	175	7D	}	CHR$(125)
126	176	7E	~	CHR$(126)

TABLE A-4 ASCII CONTROL CODES

DEC	OCT	HEX	CHAR	WHAT TO TYPE
Ø	ØØØ	ØØ	NULL	CTRL-@
1	ØØ1	Ø1	SOH	CTRL-A
2	ØØ2	Ø2	STX	CTRL-B
3	ØØ3	Ø3	ETX	CTRL-C
4	ØØ4	Ø4	ET	CTRL-D
5	ØØ5	Ø5	ENQ	CTRL-E
6	ØØ6	Ø6	ACK	CTRL-F
7	ØØ7	Ø7	BEL	CTRL-G
8	Ø1Ø	Ø8	BS	CHR$(8)
9	Ø11	Ø9	HT	CTRL-I
1Ø	Ø12	ØA	LF	CTRL-J
11	Ø13	ØB	VT	CTRL-K
12	Ø14	ØC	FF	CTRL-L
13	Ø15	ØD	CR	CTRL-M or RETURN
14	Ø16	ØE	SO	CTRL-N
15	Ø17	ØF	SI	CTRL-O
16	Ø2Ø	1Ø	DLE	CTRL-P
17	Ø21	11	DC1	CTRL-Q
18	Ø22	12	DC2	CTRL-R
19	Ø23	13	DC3	CTRL-S
2Ø	Ø24	14	DC4	CTRL-T
21	Ø25	15	NAK	CHR$(21)
22	Ø26	16	SYN	CTRL-V
23	Ø27	17	ETB	CTRL-W
24	Ø3Ø	18	CAN	CTRL-X
25	Ø31	19	EM	CTRL-Y
26	Ø32	1A	SUB	CTRL-Z
27	Ø33	1B	ESCAPE	ESC
28	Ø34	1C	FS	CHR$(28)
29	Ø35	1D	GS	CTRL-SHIFT-M
3Ø	Ø36	1E	RS	CTRL-SHIFT-N
31	Ø37	1F	US	CHR$(31)

APPENDIX B
BUS INPUT/OUTPUT SEQUENCES

This appendix illustrates the input/output sequences for the GPIB with an example of each command. The DIO mnemonics are messages defined by the IEEE-488 standard. Here are the meanings of the messages that the Apple uses. For more information see the IEEE-488 standard.

DIO Command	Hex	Binary	Translation
DCL	14	XØØ1Ø1ØØ	Device CLear
GET	Ø8	XØØØ1ØØØ	Group Execute Trigger
GTL	Ø1	XØØØØØØ1	Go To Local
LLO	11	XØØ1ØØØ1	Local LockOut
MLA		XØ1 Dev.Adr.	My (the Apple) Listen Address
MTA		X1Ø Dev.Adr.	My (the Apple) Talk Address
PPC	Ø5	XØØØØ1Ø1	Parallel Poll Configure
PPD	7Ø	X111ØØØØ	Parallel Poll Disable
PPE	6Ø	X11ØSPPP	Parallel Poll Enable
PPU	15	XØØ1Ø1Ø1	Parallel Poll Unconfigure
SDC	Ø4	XØØØØ1ØØ	Select Device Clear
SPD	19	XØØ11ØØ1	Serial Poll Disable
SPE	18	XØØ11ØØØ	Serial Poll Enable
UNL	3F	XØ111111	UNListen
UNT	5F	X1Ø11111	UNTalk

MLA and TLA are formed using the 5 lowest bits of the Apple's device address. Three mnemonics, LSTN, TALK, and XFER, are used which are not derived from the IEEE-488 standard.

LSTN — The controller is waiting to carry out a transaction with a listener. The Apple will not send new data (signalled by pulling the DAV line low) until all listeners are ready to accept data (NRFD = 1) and one listener has acknowledged the fact that there is no data currently valid (by pulling the NDAC line low).

TALK — The controller is waiting to carry out a transaction with a talker. The Apple will not accept data (by setting NDAC to 1) until the data is valid (DAV = Ø) and one listener has recognized that there is valid data on the bus (by pulling NRFD low).

XFER -- The addressed talker and listeners will carry out a handshake sequence to transfer data on the bus. The Apple does not participate.

In the commands on the next page, the character sequence <CR> represents pressing the key labelled RETURN (ASCII 13). It is not to be interpreted character by character. As before, Z$ is a variable that has been set to contain the value <CTRL-Z> (ASCII 26).

COMMAND	DIO MNEMONIC	DIO (HEX)	MGMNT	HNDSHK
PRINT "WT1";Z$;"DATA"				
or				
PRINT "WC1";Z$;"5";Z$;"DATA<CR>";				
	MTA	4Ø	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	D	44		LSTN
	A	41		LSTN
	T	54		LSTN
	A	41		LSTN
	<CR> [EOS]	ØD	EOI	LSTN
	idle	XX	ATN	
PRINT "RDAa";Z$;				
INPUT C$				
or				
PRINT "RCAa";Z$;"5";Z$;				
INPUT C$	primary talker	41	ATN	LSTN
	secondary talker	61	ATN	LSTN
	UNL	3F	ATN	LSTN
	MLA	2Ø	ATN	LSTN
	D	44		TALK
	A	41		TALK
	T	54		TALK
	A	41		TALK
	<CR>	ØD	[EOI]	TALK
	idle	XX	ATN	
PRINT "XFA";Z$;"19";Z$	primary talker	41	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	primary listener	39	ATN	LSTN
	D	44		XFER
	A	41		XFER
	T	54		XFER
	A	41	EOI	XFER
	idle	XX	ATN	
PRINT "TG1";Z$	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	GET	Ø8	ATN	LSTN
	idle	XX	ATN	

COMMAND	DIO MNEMONIC	DIO (HEX)	MGMNT	HNDSHK
PRINT "CA"	DCL	14	ATN	LSTN
	idle	XX	ATN	
PRINT "CL1";Z$	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	SDC	Ø4	ATN	LSTN
	idle	XX	ATN	
PRINT "RA"	idle	XX	ATN,REN	
PRINT "RM1";Z$	UNL	3F	ATN	LSTN
	primary listener	31	ATN,REN	LSTN
	idle	XX	ATN,REN	
PRINT "LL"	LLO	11	ATN	LSTN
	idle	XX	ATN	
PRINT "LA"	idle	XX	ATN,$\overline{\text{REN}}$	
PRINT "LO1";Z$	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	GTL	Ø1	ATN	LSTN
	idle	XX	ATN	
INPUT "SR";G$	idle	XX	[SRQ]	
PRINT "SPA";Z$;				
INPUT H$	SPE	18	SRQ,ATN	LSTN
	primary talker	41	SRQ,ATN	LSTN
	UNL	3F	SRQ,ATN	LSTN
	MLA	2Ø	SRQ,ATN	LSTN
	status byte	(4Ø)	[SRQ]	TALK
	SPD	19	[SRQ],ATN	LSTN
	idle	XX	[SRQ],ATN	

COMMAND	DIO MNEMONIC	DIO (HEX)	MGMNT	HNDSHK
INPUT "PP";I$	pp byte	(4Ø)	ATN,EOI	
	idle	XX	ATN	
PRINT "PE1A2B"	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	PPC	Ø5	ATN	LSTN
	PPE + SPPP *	61	ATN	LSTN
	primary listener	32	ATN	LSTN
	PPC	Ø5	ATN	LSTN
	PPE + SPPP *	62	ATN	LSTN
	idle	XX	ATN	
PRINT "PD1";Z$	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	PPC	Ø5	ATN	LSTN
	PPD	7Ø	ATN	LSTN
	idle	XX	ATN	
PRINT "PU"	PPU	15	ATN	LSTN
	idle	XX	ATN	
PRINT "AB"	idle	XX	IFC	
PRINT "LF1"				
PRINT "WT1";Z$;"DATA"	MTA	4Ø	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	D	44		LSTN
	A	41		LSTN
	T	54		LSTN
	A	41		LSTN
	<CR>	ØD		LSTN
	<LF>	ØA	EOI	LSTN
	idle	XX	ATN	

* SPPP refers to the Sense bit and the 3 bit assignment bits. The code for PPE is logically ORed with these bits. Refer to the section on the PE command for more details.

COMMAND	DIO MNEMONIC	DIO (HEX)	MGMNT	HNDSHK
PRINT "ES!"				
PRINT "WT1";Z$;"DATA!";	MTA	4Ø	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	D	44		LSTN
	A	41		LSTN
	T	54		LSTN
	A	41		LSTN
	!	21	EOI	LSTN
	idle	XX	ATN	
PRINT "SC1"	idle	XX	ATN	
PRINT "DV16"				
PRINT "WT1";Z$;"DATA"	MTA	5Ø	ATN	LSTN
	UNL	3F	ATN	LSTN
	primary listener	31	ATN	LSTN
	D	44		LSTN
	A	41		LSTN
	T	54		LSTN
	A	41		LSTN
	<CR>	ØD	EOI	LSTN
	idle	XX	ATN	
PRINT "UT"	UNT	5F	ATN	LSTN
	idle	XX	ATN	

APPENDIX C
9914 REGISTERS

You will need to use the information in this appendix only if you want to send information directly to the registers contained in the TMS 9914 chip on the IEEE-488 card. Details on the use of the 9914 are beyond the scope of this document. Please refer to the 9914 specification sheets if you want or need more information.

RS2, RS1 and RSØ are the register address lines on the 9914. They are connected to the three low address lines on the Apple bus.

The N in the Apple addresses stands for eight plus the number of the slot into which the GPIB is inserted. For example, if the GPIB is installed in slot 3 then the address of the Auxiliary Command Register is CØB3 or CØBB. Because address bit 3 is not decoded, two addresses can access each of the 9914 registers.

9914 REGISTER				APPLE ADDRESS	
			READ		
RS2	RS1	RSØ			
Ø	Ø	0	INT STATUS Ø	CØNØ	CØN8
Ø	Ø	1	INT STATUS 1	CØN1	CØN9
Ø	1	Ø	ADDRESS STATUS	CØN2	CØNA
Ø	1	1	BUS STATUS	CØN3	CØNB
1	Ø	Ø	ADDRESS SWITCH 1	CØN4	CØNC
1	1	Ø	CMD PASS THROUGH	CØN6	CØNE
1	1	1	DATA IN	CØN7	CØNF
			WRITE		
RS2	RS1	RSØ			
Ø	Ø	Ø	INT MASK Ø	CØNØ	CØN8
Ø	Ø	1	INT MASK 1	CØN1	CØN9
Ø	1	1	AUXILIARY CMD	CØN3	CØNB
1	Ø	Ø	ADDRESS REG	CØN4	CØNC
1	Ø	1	SERIAL POLL	CØN5	CØND
1	1	Ø	PARALLEL POLL	CØN6	CØNE
1	1	1	DATA OUT	CØN7	CØNF

NOTE: The 65Ø2 performs a read operation before each write operation. Since some registers on the 9914 are reset by a read, you must take care that the proper action is taken. Refer to a 65Ø2 Programming Manual and the 9914 specifications for further details.

An example of the use of some of the registers in the 9914 can be found in Appendix E.

APPENDIX D
PROGRAMMING AIDS

This appendix contains a series of helpful subroutines and programs for learning about and using the Apple IEEE-488 Interface Card.

PROGRAM 1: CHR$ FUNCTION FOR INTEGER BASIC

Purpose: This function gives you the ability to generate those characters that cannot be sent by the Apple's keyboard (see Tables A-3 and A-4 in Appendix A). It should be used with Integer BASIC in place of the CHR$ function available to APPLESOFT.

Setup: Place the following lines of code somewhere in your program. Of course you may change the line numbers, but don't change anything else!! Line 12Ø POKEs the proper ASCII code right between the " marks in the statement CHR$="A".

Code:

```
11Ø   CHS = CHR +128* (CHR<128)

12Ø   LC1= PEEK (224): LC2= PEEK (
      225)-(LC1>243): POKE 79+LC1-
      256*(LC2>127)+(LC2-255*(LC1>
      127))*256,CHS:CHR$="A":RETURN
```

Use: 1) Place the ASCII code of the desired character into CHR.
2) GOSUB 11Ø (or the line number of your first line).
3) The value returned in CHR$ is the character you want!

Errors: If line 12Ø is used in a long program on an Apple with more than 32K bytes of memory, there is a very remote chance that you will get a *** >32767 ERR. If this should happen, insert a REM statement with about 8Ø characters on a higher line number to force the CHR$ function down in memory.

Source: Contributed Programs Volumes 3-5 (contains an explanation!)
Apple Product #A2L0014

PROGRAM 2: ONERR ROUTINE

Purpose: The ONERR..GOTO statement can be used to detect and recover from the BUS error. Since the BUS error cannot occur during normal bus protocol, it indicates that an abnormal situation exists; for example, it may occur if the bus is not connected properly.

Code:

```
5  DIM N$(5)
1Ø  D$ = CHR$ (4)                :REM  CTRL-D = DOS COMMAND
2Ø  Z$ = CHR$ (26)               :REM  CTRL-Z = BUS COMMAND
25  PRINT
3Ø  ONERR GOTO 5ØØ               :REM  ONERR ROUTINE AT 5ØØ
4Ø  PRINT D$;"PR#3"              :REM  ENABLE OUTPUT TO BUS
5Ø  PRINT D$;"IN#3"              :REM  ENABLE INPUT FROM BUS
55  PRINT "SC1"                  :REM  SET SCREEN TO ON
6Ø  PRINT "WT1"; CHR$ (243); Z$;"ONERR GOTO TEST"
61                                REM  DO A NORMAL WRITE TO LISTENER 1
62                                REM  ERROR IF BUS NOT CONNECTED
7Ø  PRINT D$;"PR#Ø"              :REM  ENABLE OUTPUT TO SCREEN
8Ø  PRINT D$;"IN#Ø"              :REM  ENABLE INPUT FROM KEYBOARD
9Ø  END
5ØØ  PRINT : PRINT               :REM  ONERR ROUTINE
51Ø  PRINT "'ONERR GOTO' GOT US HERE"
511  PRINT : PRINT "ERROR FLAG = ";  :REM SHOULD BE 49
512  PRINT PEEK (222) : PRINT    :REM  PRINT ERROR NUMBER
513  PRINT "ERROR OCCURRED AT LINE ";
514  PRINT PEEK (218) + 256 * PEEK (219)
515                               REM  USE DOS ONERR LOCATIONS
516                               REM  TO PRINT LINE NUMBER
52Ø  POKE 216,Ø                  :REM  RESET ONERR FLAG
53Ø  PRINT : PRINT
54Ø  INPUT "FIX BUS THEN HIT RETURN TO CONTINUE";N$
55Ø  GOTO 25
```

Use: To test the ONERR routine:

1) Enter the program.
2) Make sure the bus is disconnected.
3) Run the program.
4) Reconnect the bus and continue.

Note: A good discussion of ONERR can be found in the DOS 3.2 Manual, Apple Product #A2LØØ12.

PROGRAM 3: SKIP REINITIALIZE FUNCTION

Purpose: When a PR#n and IN#n are issued with the IEEE-488 card in slot n, all the card's options are initialized to their default values. The subroutine beginning at line 2ØØØ does a PR#3, IN#3 without reinitializing the card. It can be adapted to any slot.

Code:

```
1ØØ  DIM A$(1Ø)                  :REM  DELETE LINES 1ØØ-12Ø
11Ø  DIM C$(1Ø)                  :REM   IF YOU ARE USING
12Ø  DIM G$(1Ø)                  :REM    APPLESOFT.
13Ø  Z$ = ""                     :REM Z$ CONTAINS CTRL-Z
14Ø  D$ = ""                     :REM D$ CONTAINS CTRL-D
1ØØØ  PRINT D$;"IN#3"            :REM ENABLE INPUT FROM SLOT 3
1Ø1Ø  PRINT D$;"PR#3"            :REM ENABLE OUTPUT TO SLOT 3
1Ø2Ø  PRINT "SC1"                :REM TURN ON THE SCREEN
1Ø3Ø  PRINT "WT123";Z$;"HI"      :REM SEND "HI" TO THE GPIB & SCREEN
1Ø5Ø  INPUT "SR";G$              :REM SERVICE REQUESTED?
1Ø6Ø  PRINT D$;"IN#Ø"            :REM ENABLE INPUT FROM KEYBOARD
1Ø7Ø  PRINT D$;"PR#Ø"            :REM ENABLE OUTPUT TO SCREEN ONLY
1Ø8Ø  INPUT "THIS LINE DOES NOT APPEAR ON THE BUS -- PRESS RETURN TO
    CONTINUE.";A$
1Ø9Ø  GOSUB 2ØØØ                 :REM RESTORE INPUTS AND OUTPUTS
11ØØ  PRINT "WC123";Z$;"5";Z$;"-DATA";  :REM SEND "-DATA" TO
                                        :REM   GPIB AND SCREEN
111Ø  PRINT "RDA";Z$;            :REM SEND COMMAND, ADRS, CTRL-Z
112Ø  INPUT C$                   :REM AND GET INPUT STRING
113Ø  PRINT D$;"IN#Ø"            :REM ENABLE INPUT FROM KEYBOARD
114Ø  PRINT D$;"PR#Ø"            :REM ENABLE OUTPUT TO SCREEN
12ØØ  END
1998  REM ---- THIS SUBROUTINE DOES A PR#3/IN#3 WITHOUT RESETTING
1999  REM ---- THE GPIB DEFAULT VALUES.
2ØØØ  POKE 54,6                  :REM POKE ADDRESS OF CHARACTER
2Ø1Ø  POKE 55,195                :REM   OUTPUT SUBROUTINE.
                                 :REM     195 = 192 + SLOT NUMBER
2Ø2Ø  POKE 56,3                  :REM POKE ADDRESS OF CHARACTER
2Ø3Ø  POKE 57,195                :REM   INPUT SUBROUTINE.
                                 :REM     195 = 192 + SLOT NUMBER
2Ø4Ø  CALL 1ØØ2                  :REM CALL DOS INITILIZATION
2Ø5Ø  RETURN
```

Use:
1) Type in the code.
2) If you don't have DOS, delete line 2Ø4Ø and the D$'s.
3) Run it.
4) You can use lines 2ØØØ-2Ø5Ø as a general subroutine.

Errors: This code will not work with DOS 3.1.

Notes:
1) DOS input and output locations are discussed in the chapter on Input and Output in the DOS 3.2 (or 3.3) Manual.
2) Lines 1ØØ-12Ø should only be used with Integer BASIC.

PROGRAM 4: INPUT WITH COMMAS

Purpose: APPLESOFT interprets a comma in a data string as the end of input. Use this routine to input data from a device that sends commas. Input terminates when a CR is read. (Note that you can easily modify it to terminate on EOS as well.)

Setup: Type in the lines of code

Code:

```
5  REM   CAN ONLY BE USED IN APPLESOFT
6  REM   INPUT STRING IS STORED IN B$
7  REM   INPUT IS TERMINATED WITH CR
25 A$ = "" : B$ = ""              :REM STRINGS INITIALLY EMPTY
30  GET A$                        :REM READ A CHARACTER AND
40  PRINT A$;                     :REM ECHO IT. SEE NOTE.
50  IF ASC (A$) = 13 THEN RETURN:REM RETURN ON CR
60 B$ = B$ + A$                   :REM ADD CHARACTER TO STRING
70  GOTO 30
```

Use: 1) Call with GOSUB 25 (or whatever you call that line)
2) Input string will be returned in B$

Notes: 1) The firmware expects strings to be read from the bus using the INPUT statement. Since the INPUT statement echoes characters to the screen, a GET statement must be followed by a PRINT statement to simulate this echo. In this way you can fool the firmware into thinking that normal input is being done.

Examples: This routine is used in the second sample program in Appendix G. Refer to that example to see how it can be used.

APPENDIX E
FAST TRANSFER ROUTINES

Purpose: These routines are an example of how one can do fast block transfers over the GPIB. These routines communicate directly with the 9914, as opposed to normal GPIB transfers, which send a string to DOS, which uses the Apple Monitor Input or Output Registers to transfer the string character by character to the IEEE-488 card, which finally sends it to the 9914. These routines are also an example of how a machine language subroutine can be used to implement one of the GPIB instructions. This will be useful if you want to use the GPIB with the Pascal System. See Appendix F for more details on how to use the GPIB with Pascal.

Background: These routines were generated using the Apple 65Ø2 Assembler/Editor. The code is ORG'd at $5ØØØ and the data at $52ØØ. You will probably need to change these locations depending on your application. The code assumes that your card is in slot 3, and that your EOS character is CR. (See the declarations of EOS and SLOT at $52ØØ and $52Ø2.) The end of this Appendix explains how these routines can be run and tested.

GPIB SUBR. DEMO

This BASIC test program may be used to call the machine code subroutines. It first POKEs an RTS in place of the BRK at location 21569 ($5441); the BRK is used for test purposes. It then calls the test routine at $54ØØ, which in turn calls the other routines. Finally, it executes some GPIB instructions to show that the routines don't interfere with normal GPIB operation.

```
1Ø  DIM S$(3Ø)          : REM  DELETE THIS LINE FOR APPLESOFT
15  POKE 21569,96       : REM  PUT AN RTS IN PLACE OF BRK
2Ø  CALL 215Ø4          : REM  $54ØØ IS THE TEST PROGRAM
3Ø  Z$ = ""             : REM  Z$ IS CTRL-Z, GPIB COMMAND
4Ø  D$ = ""             : REM  D$ IS CTRL-D, DOS COMMAND
5Ø  PRINT D$;"PR#3"     : REM  ENABLE OUTPUT TO CARD
6Ø  PRINT D$;"IN#3"     : REM  ENABLE INPUT FROM CARD
65  PRINT "SC1"         : REM  TURN SCREEN ON
7Ø  PRINT "RDB";Z$;     : REM  READ FROM TALKER "B"
75  INPUT S$
8Ø  PRINT D$;"PR#Ø"     : REM  ENABLE OUTPUT TO SCREEN
9Ø  PRINT D$;"IN#Ø"     : REM  ENABLE INPUT FROM KEYBOARD
1ØØ  PRINT "S$ = ";S$   : REM  ECHO INPUT STRING
11Ø  PRINT D$;"PR#3"    : REM  ENABLE OUTPUT TO CARD
12Ø  PRINT D$;"IN#3"    : REM  ENABLE INPUT FROM CARD
125  PRINT "SC1"        : REM  TURN SCREEN ON
13Ø  PRINT "WT1";Z$;"FIRMWARE"  : REM  SEND "FIRMWARE" TO "1"
14Ø  PRINT D$;"PR#Ø"    : REM  ENABLE OUTPUT TO SCREEN
15Ø  PRINT D$;"IN#Ø"    : REM  ENABLE INPUT FROM KEYBOARD
2ØØ  END
```

SUBROUTINES

All of the following routines have arbitrary starting addresses. Relocate them according to your system configuration. You must also relocate the seven bytes of storage. The location of the buffers for data and device lists is up to you.

GENERAL CONTROL VALUES

The general control values assume that the Apple is device #Ø.

```
C080:           19 GPIB     EQU  $C080     ;SET BASE ADDRESS OF ROM
0000:           20 ;                         OFFSET FOR SLOT 3 ASSUMED
C080:           21 INT0     EQU  GPIB+0    ;INTERRUPT REG. 0
C080:           22 INTM0    EQU  GPIB+0    ;INTRP MASK REG 0
0020:           23 BIM      EQU  $20       ;BYTE IN MASK
0010:           24 BOM      EQU  $10       ;BYTE OUT MASK
0008:           25 EOIMK    EQU  $08       ;EOI MASK
0000:           26 ;
C081:           27 INTM1    EQU  GPIB+1    ; INTRP MASK REG 1
0000:           28 ;
C083:           29 AUXCMD   EQU  GPIB+3    ;AUXILIARY COMMAND REGISTER
0080:           30 RESET    EQU  $80       ;SOFTWARE CHIP RESET
0000:           31 RSTCLR   EQU  $00       ;STOP SOFTWARE RESET
0083:           32 HDFA     EQU  $83       ;HOLDOFF ON ALL DATA
0003:           33 HDACLR   EQU  $03       ;CLEAR HOLDOFF ON ALL
0089:           34 LON      EQU  $89       ;LISTEN ONLY
008A:           35 TON      EQU  $8A       ;TALK ONLY
008F:           36 SIC      EQU  $8F       ;SEND INTERFACE CLEAR
000F:           37 SICLR    EQU  $0F       ;CLEAR SIC
0002:           38 RHDF     EQU  $02       ;RELEASE RFD HOLDOFF
0008:           39 FEOI     EQU  $08       ;SEND EOI WITH NEXT BYTE
000B:           40 GTS      EQU  $0B       ;GO TO STANDBY
000C:           41 TCA      EQU  $0C       ;TAKE CONTROL ASYNCHRONOUSLY
000D:           42 TCS      EQU  $0D       ;TAKE CONTROL SYNCHRONOUSLY
0000:           43 ;
C087:           44 DIN      EQU  GPIB+7    ;DATA IN REGISTER
C087:           45 DOUT     EQU  GPIB+7    ;DATA OUT REGISTER
0000:           46 ;
0000:           47 ; GPIB COMMANDS
0000:           48 ;
0020:           49 MLA      EQU  $20       ;MY LISTEN ADDRESS
0040:           50 MTA      EQU  $40       ;MY TALK ADDRESS
003F:           51 UNL      EQU  $3F       ;UNIVERSAL UNLISTEN
```

INITIALIZE 9914

All of the routines make common assumptions about the state of the 9914 chip upon entry, and all will leave the 9914 in this same state upon exit. The INIT routine sets the 9914 to this desired state. These assumptions are:

- BO set
- All interrupts masked off
- TON (talk only) on
- Not listener active (LA)
- No holdoffs enabled or active
- No 65Ø2 registers destroyed
- Status obtained by polling the 9914

SUBROUTINES CALLED: WAIT

```
5ØØØ:           1Ø1          ORG  $5ØØØ
5ØØØ:           1Ø2 ;
5ØØØ:Ø8         1Ø3 INIT:    PHP             ;SAVE REGISTERS
5ØØ1:48         1Ø4          PHA
5ØØ2:98         1Ø5          TYA
5ØØ3:48         1Ø6          PHA
5ØØ4:8A         1Ø7          TXA
5ØØ5:48         1Ø8          PHA
5ØØ6:AC Ø2 52   1Ø9          LDY  SLOT       ;GPIB SLOT INDICATOR
5ØØ9:A9 8Ø      11Ø          LDA  #RESET     ;RESET 9914
5ØØB:99 83 CØ   111          STA  AUXCMD,Y
5ØØE:A9 ØØ      112          LDA  #RSTCLR    ;TURN OFF RESET
5Ø1Ø:99 8Ø CØ   113          STA  INTMØ,Y    ;CLEAR INTERRUPTS
5Ø13:99 81 CØ   114          STA  INTM1,Y    ;CLEAR INTERRUPTS
5Ø16:99 83 CØ   115          STA  AUXCMD,Y
5Ø19:A9 8F      116          LDA  #SIC       ;SEND INTERFACE CLEAR
5Ø1B:99 83 CØ   117          STA  AUXCMD,Y   ;AND TAKE CONTROL
5Ø1E:A2 Ø4      118          LDX  #4         ;5.12 MSEC DELAY
5Ø2Ø:AØ ØØ      119          LDY  #Ø         ;INITIALIZE INNER LOOP
5Ø22:88         12Ø INIT1:   DEY             ;5 USEC INNER LOOP
5Ø23:DØ FD      121          BNE  INIT1      ;128Ø USEC TOTAL
5Ø25:CA         122          DEX
5Ø26:DØ FA      123          BNE  INIT1
5Ø28:AC Ø2 52   124          LDY  SLOT
5Ø2B:A9 ØF      125          LDA  #SICLR     ;CLEAR INTERFACE CLR
5Ø2D:99 83 CØ   126          STA  AUXCMD,Y
5Ø3Ø:A9 8A      127          LDA  #TON       ;SET TALK ONLY MODE
5Ø32:99 83 CØ   128          STA  AUXCMD,Y
5Ø35:2Ø 32 51   129          JSR  WAIT       ;WAIT FOR BYTE OUT
5Ø38:           13Ø ;                          TO INDICATE CONTROL TAKEN
5Ø38:68         131          PLA             ;RESTORE REGISTERS
5Ø39:AA         132          TAX
5Ø3A:68         133          PLA
5Ø3B:A8         134          TAY
5Ø3C:68         135          PLA
5Ø3D:28         136          PLP
5Ø3E:6Ø         137          RTS             ;RETURN TO CALLING POINT
```

SEND ROUTINE

The send routine can be used in place of the WT or WC command. It terminates either when COUNT bytes have been sent or when EOS is sent. It asserts EOI in both cases.

```
     INPUTS:   ABUF = Pointer to the listener list
               DBUF = Pointer to the data buffer
               COUNT = Number of bytes to send; Ø causes no data.
               EOS = The terminating character
     OUTPUTS:  None to software
     SUBROUTINES CALLED:   WAIT
     REGISTERS DESTROYED:  None

503F:Ø8         157 SEND:   PHP             SAVE REGISTERS
5Ø4Ø:48         158         PHA
5Ø41:98         159         TYA
5Ø42:48         16Ø         PHA
5Ø43:8A         161         TXA
5Ø44:48         162         PHA
5Ø45:AC Ø2 52   163         LDY  SLOT       ;GPIB SLOT INDICATOR
5Ø48:A9 4Ø      164         LDA  #MTA       ;MY TALK ADDRESS
5Ø4A:99 87 CØ   165         STA  DOUT,Y
5Ø4D:2Ø 32 51   166         JSR  WAIT       ;WAIT FOR BYTE OUT
5Ø5Ø:A9 3F      167         LDA  #UNL       ;UNIVERSAL UNLISTEN
5Ø52:99 87 CØ   168         STA  DOUT,Y
5Ø55:A2 ØØ      169         LDX  #Ø         ;BUFFER INDEX
5Ø57:BD Ø3 52   17Ø SEND1:  LDA  ABUF,X     ;GET LISTENER ADDRESS
5Ø5A:C9 2Ø      171         CMP  #$2Ø       ;LOWER LIMIT OF RANGE
5Ø5C:3Ø 1Ø      172         BMI  SEND2
5Ø5E:C9 3F      173         CMP  #$3F       ;UPPER LIMIT
5Ø6Ø:1Ø ØC      174         BPL  SEND2
5Ø62:2Ø 32 51   175         JSR  WAIT       ;WAIT FOR UNL BYTE OUT
5Ø65:BD Ø3 52   176         LDA  ABUF,X     ;GET LISTENER AGAIN
5Ø68:99 87 CØ   177         STA  DOUT,Y     ;OUTPUT IT TO GPIB
5Ø6B:E8         178         INX
5Ø6C:1Ø E9      179         BPL  SEND1      ;ALWAYS TAKEN
5Ø6E:           18Ø ;                         (MAX OF 31 LISTENERS!)
5Ø6E:A9 ØB      181 SEND2:  LDA  #GTS       ;GO TO STANDBY
5Ø7Ø:99 83 CØ   182         STA  AUXCMD,Y
5Ø73:2Ø 32 51   183         JSR  WAIT       ;WAIT FOR ATN TO BE
5Ø76:           184 ;                         DE-ASSERTED
5Ø76:A2 ØØ      185         LDX  #Ø         ;INITIALIZE DATA BUF INDEX
5Ø78:EC Ø1 52   186         CPX  COUNT      ;IS INITIAL COUNT =Ø?
5Ø7B:FØ 28      187         BEQ  SEND6      ;YES--GO QUIT
5Ø7D:CE Ø1 52   188 SEND3:  DEC  COUNT
5Ø8Ø:FØ 11      189         BEQ  SEND5      ;IF COUNT FINISHED GO SEND EOI
5Ø82:           19Ø ;                         WITH LAST CHARACTER
5Ø82:BD Ø5 52   191         LDA  DBUF,X     ;GET DATA BYTE
5Ø85:CD ØØ 52   192         CMP  EOS        ;IS IT LAST CHARACTER?
5Ø88:FØ Ø9      193         BEQ  SEND5      ;YES--GO HANDLE IT
5Ø8A:99 87 CØ   194         STA  DOUT,Y     ;OUTPUT DATA TO GPIB
5Ø8D:2Ø 32 51   195 SEND4:  JSR  WAIT       ;WAIT FOR DATA BYTE OUT
5Ø9Ø:E8         196         INX
5Ø91:DØ EA      197         BNE  SEND3      ;ALWAYS TAKEN
```

```
5Ø93:A9 Ø8      198 SEND5:  LDA  #FEOI      ;FORCE EOI W/ EOS
5Ø95:99 83 CØ   199         STA  AUXCMD,Y
5Ø98:BD Ø5 52   2ØØ         LDA  DBUF,X     ;GET BYTE AGAIN
5Ø9B:99 87 CØ   2Ø1         STA  DOUT,Y     ;OUTPUT TO GPIB
5Ø9E:E8         2Ø2         INX
5Ø9F:CE Ø1 52   2Ø3         DEC  COUNT      ;FOR CONSISTENCY
5ØA2:2Ø 32 51   2Ø4         JSR  WAIT       ;WAIT FOR LAST BYTE OUT
5ØA5:A9 ØC      2Ø5 SEND6:  LDA  #TCA       ;TAKE CONTROL
5ØA7:99 83 CØ   2Ø6         STA  AUXCMD,Y
5ØAA:2Ø 32 51   2Ø7         JSR  WAIT       ;WAIT FOR CONTROL
5ØAD:68         2Ø8         PLA             ;RESTORE REGISTERS
5ØAE:AA         2Ø9         TAX
5ØAF:68         21Ø         PLA
5ØBØ:A8         211         TAY
5ØB1:68         212         PLA
5ØB2:28         213         PLP
5ØB3:6Ø         214         RTS             ;RETURN TO CALL POINT
```

RECEIVE ROUTINE

The receive routine can be used in place of the RD or RC command. It terminates either when COUNT bytes have been read or when EOS is read or when EOI goes high.

```
INPUTS:    ABUF = Pointer to the talker list
           DBUF = Pointer to the data buffer
           COUNT = Number of bytes to receive; Ø receives no data
           EOS = The terminating character
OUTPUTS:   Data buffer is filled
SUBROUTINES CALLED:   WAIT
REGISTERS DESTROYED:  None
```

```
5ØB4:Ø8         235 RECV:   PHP             ;SAVE REGISTERS
5ØB5:48         236         PHA
5ØB6:98         237         TYA
5ØB7:48         238         PHA
5ØB8:8A         239         TXA
5ØB9:48         24Ø         PHA
5ØBA:AC Ø2 52   241         LDY  SLOT       ;GPIB SLOT IDENTIFIER
5ØBD:A2 ØØ      242         LDX  #Ø
5ØBF:AD Ø3 52   243         LDA  ABUF       ;GET TALKER DEVICE NO.
5ØC2:99 87 CØ   244         STA  DOUT,Y     ;OUTPUT IT TO GPIB
5ØC5:2Ø 32 51   245         JSR  WAIT       ;WAIT FOR BYTE OUT
5ØC8:A9 3F      246         LDA  #UNL       ;UNIVERSAL UNLISTEN
5ØCA:99 87 CØ   247         STA  DOUT,Y     ;OUTPUT IT TO GPIB
5ØCD:2Ø 32 51   248         JSR  WAIT       ;WAIT FOR BYTE OUT
5ØDØ:A9 2Ø      249         LDA  #MLA       ;MY LISTEN ADDRESS
5ØD2:99 87 CØ   25Ø         STA  DOUT,Y     ;OUTPUT IT TO GPIB
5ØD5:A9 83      251         LDA  #HDFA      ;ENABLE HOLDOFF ON ALL
5ØD7:99 83 CØ   252         STA  AUXCMD,Y
5ØDA:A9 89      253         LDA  #LON       ;LISTEN ONLY-CLEAR TON
5ØDC:99 83 CØ   254         STA  AUXCMD,Y
5ØDF:A9 ØB      255         LDA  #GTS       ;GO TO STANDBY
5ØE1:99 83 CØ   256         STA  AUXCMD,Y
5ØE4:2Ø 32 51   257         JSR  WAIT       ;WAIT FOR MLA BYTE OUT
```

```
50E7:B9 80 C0  258 RECV1:  LDA  INT0,Y    ;WAIT FOR BYTE IN
50EA:29 28     259         AND  #EOIMK+BIM ;TEST FOR EOI OR BI
50EC:F0 F9     260         BEQ  RECV1
50EE:29 08     261         AND  #EOIMK     ;IS IT EOI?
50F0:D0 18     262         BNE  RECV2      ;YES
50F2:B9 87 C0  263         LDA  DIN,Y      ;GET DATA BYTE FROM GPIB
50F5:9D 05 52  264         STA  DBUF,X     ;STORE IT IN BUFFER
50F8:CD 00 52  265         CMP  EOS        ;IS IT THE EOS CHAR?
50FB:F0 14     266         BEQ  RECV3      ;YES
50FD:E8        267         INX
50FE:A9 02     268         LDA  #RHDF      ;RELEASE HOLDOFF
5100:99 83 C0  269         STA  AUXCMD,Y
5103:CE 01 52  270         DEC  COUNT
5106:F0 0C     271         BEQ  RECV4      ;GO IF COUNT =0
5108:D0 DD     272         BNE  RECV1      ;ALWAYS TAKEN--GO FOR MORE
510A:B9 87 C0  273 RECV2:  LDA  DIN,Y      ;GET LAST BYTE
510D:9D 05 52  274         STA  DBUF,X
5110:E8        275         INX
5111:CE 01 52  276 RECV3:  DEC  COUNT
5114:A9 0D     277 RECV4:  LDA  #TCS       ;TAKE CONTROL SYNC
5116:99 83 C0  278         STA  AUXCMD,Y
5119:20 32 51  279         JSR  WAIT       ;WAIT FOR CONTROL
511C:A9 02     280         LDA  #RHDF      ;RELEASE HOLDOFF
511E:99 83 C0  281         STA  AUXCMD,Y
5121:A9 8A     282         LDA  #TON       ;SET TALK ONLY, CLR LON
5123:99 83 C0  283         STA  AUXCMD,Y
5126:A9 03     284         LDA  #HDACLR    ;CLEAR HOLDOFF MODE
5128:99 83 C0  285         STA  AUXCMD,Y
512B:68        286         PLA             ;RESTORE REGISTERS
512C:AA        287         TAX
512D:68        288         PLA
512E:A8        289         TAY
512F:68        290         PLA
5130:28        291         PLP
5131:60        292         RTS             ;RETURN TO CALL POINT
```

WAIT ROUTINE

The wait routine reads the INT0 register until the Byte Out or Byte In bit is set.

```
INPUTS:  Register Y contains the slot number
OUTPUTS: None
PROCEDURES CALLED:    None
REGISTERS DESTROYED: None
```

```
5132:B9 80 C0  310 WAIT:   LDA  INT0,Y    ;GET BYTE OUT FLAG
5135:29 10     311         AND  #BOM
5137:F0 F9     312         BEQ  WAIT       ;LOOP UNTIL SET
5139:60        313         RTS
```

DATA BUFFERS

The following data buffers work with the test routine given. You may have to change them to suit your purposes. Notice that the default value of EOS is CR, the default number of bytes counted is 255, and the default slot is 3. ABUF, the list of listeners or talkers is only large enough to contain one listener or talker; this may be set according to your needs. This list may actually end with any non-listener or non-talker depending on which routine is using it.

```
5200:           325          ORG  $5200
5200:           326 ;
5200:0D         327 EOS      DFB  $0D        ;EOS CHARACTER--DEFAULT IS CR
5201:FF         328 COUNT    DFB  $FF        ;BYTE COUNT FOR DATA BUFFER
5202:           329 ;                          --DEFAULT IS 255
5202:30         330 SLOT     DFB  $3         ;OFFSET TO SLOT 3 ADDRESSES
5203:           331 ;                          --DEFAULT IS SLOT #3
5203:           332 ABUF     DS   2          ;DEVICE LIST
5205:           333 ;                          --MUST END WITH $FF BYTE
5205:           334 DBUF     DS   256        ;DATA BUFFER
```

SUBROUTINE TEST

This routine tests the subroutines that have already been given. It is called by the GPIB SUBR DEMO given at the beginning of this section. It sends data to listener $21 (ASCII "!") and receives data from talker $41 (ASCII "A"). The data buffer (DBUF) is filled with the numbers 1 through 5 followed by a CR. This data is first sent with COUNT as initialized (255) so that the CR terminates output. Then it is sent with a COUNT of 5; thus the CR doesn't get sent. The BRK in line 376 was replaced by an RTS instruction, thus the RECV routine is only called once, and the loop is never executed.

Notes: The code assumes that <EOS> is set to <CR>. If you have changed your EOS, the first call to SEND will send 255 bytes.

```
5400:           347          ORG  $5400
5400:           348 ;
5400:20 00 50   349          JSR  INIT       ;SET UP GPIB INITIALLY
5403:EA         350          NOP
5404:A9 21      351          LDA  #$21       ;LISTENER ADDRESS
5406:8D 03 52   352          STA  ABUF
5409:A9 FF      353          LDA  #$FF       ;LIST END
540B:8D 04 52   354          STA  ABUF+1
540E:A9 01      355          LDA  #1
5410:8D 05 52   356          STA  DBUF       ;DATA TO BE SENT
5413:A9 02      357          LDA  #2
5415:8D 06 52   358          STA  DBUF+1
5418:A9 03      359          LDA  #3
541A:8D 07 52   360          STA  DBUF+2
541D:A9 04      361          LDA  #4
541F:8D 08 52   362          STA  DBUF+3
5422:A9 05      363          LDA  #5
5424:8D 09 52   364          STA  DBUF+4
```

```
5427:A9 ØD      365          LDA  #$ØD      ;6 BYTES OF DATA INCL CR
5429:8D ØA 52   366          STA  DBUF+5
542C:2Ø 3F 5Ø   367          JSR  SEND
542F:EA         368          NOP
5430:A9 Ø5      369          LDA  #5        ;5 BYTES OF DATA, NO CR
5432:8D Ø1 52   37Ø          STA  COUNT
5435:2Ø 3F 5Ø   371          JSR  SEND
5438:EA         372          NOP
5439:A9 41      373          LDA  #$41      ;TALKER ADDRESS
543B:8D Ø3 52   374          STA  ABUF
543E:2Ø B4 5Ø   375 LOOP:    JSR  RECV
5441:ØØ         376          BRK
5442:4C 3E 54   377          JMP  LOOP
5445:           378 ;
5445:           379 ;END
```

SYMBOL TABLE

```
 52Ø3 ABUF          CØ83 AUXCMD          2Ø BIM          1Ø BOM
 52Ø1 COUNT         52Ø5 DBUF         CØ87 DIN       CØ87 DOUT
   Ø8 EOIMK         52ØØ EOS            Ø8 FEOI      CØ8Ø GPIB
   ØB GTS             Ø3 HDACLR         83 HDFA      5ØØØ INIT
 5Ø22 INIT1         CØ8Ø INTØ         CØ8Ø INTMØ      CØ81 INTM1
   89 LON           543E LOOP           2Ø MLA         4Ø MTA
 51ØA RECV2         5ØB4 RECV         5ØE7 RECV1      5111 RECV3
 5114 RECV4           8Ø RESET          Ø2 RHDF         ØØ RSTCLR
 5Ø57 SEND1         5Ø6E SEND2          5Ø3F SEND     5Ø7D SEND3
?5Ø8D SEND4         5Ø93 SEND5          5ØA5 SEND6      ØF SICLR
   8F SIC           52Ø2 SLOT           ØC TCA          ØD TCS
   8A TON             3F UNL          5132 WAIT
```

DATA RATES

When the GPIB does a WRITE operation, it transfers the data at a rate of 1.5 kilobytes per second. This does not include the roughly 3.5 millisecond interval during which processing of the instruction takes place. For a READ operation, the GPIB transfers data at a rate of about 1 kilobyte per second.

The performance of the IEEE-488 bus can be improved with the preceding routines. These routines, as listed, send and receive data at a rate of 2Ø kilobytes per second. Even this is not optimal. If the above routines are streamlined to transfer data by count only, they will send data at a rate of 5Ø kilobytes per second.

RUNNING AND TESTING

Our version of the APPLESOFT program used to run these subroutines was called GPIB SUBR DEMO. The assembly language routine source code was saved in the file GPIB SUBR. The 65Ø2 Assembler places the assembled code into the files:

```
GPIB SUBR.OBJØ  (routines)
GPIB SUBR.OBJ1  (data)
GPIB SUBR.OBJ2  (test code)
```

The routines may be tested with the following sequence of commands:

```
] BLOAD GPIB SUBR.OBJØ
] BLOAD GPIB SUBR.OBJ1
] BLOAD GPIB SUBR.OBJ2
] RUN GPIB SUBR DEMO
```

We recommend that you use a ZT 488 GPIB Analyzer to verify correct operation of the routines. The sequence of bus signals should be:

DIO	MGMNT	DIO	MGMNT	DIO	MGMNT	DIO	MGMNT	DIO	MGMNT
4Ø	ATN	4Ø	ATN	41	ATN	42	ATN	4Ø	ATN
3F	ATN	3F	ATN	3F	ATN	3F	ATN	3F	ATN
21	ATN	21	ATN	2Ø	ATN	2Ø	ATN	31	ATN
1		1		INPUT		INPUT		46	
2		2		any		any		49	
3		3		sequence		sequence		52	
4		4		terminated		terminated		4D	
5		5	EOI	with		with		57	
ØD	EOI			ØD or EOI		ØD or EOI		41	
								52	
								45	
								ØD	EOI

APPENDIX F
PROGRAMMING IN PASCAL

Because the GPIB firmware is designed to interact with BASIC, the Monitor and DOS, it will not behave properly if commands are sent to the GPIB using the Pascal WRITE and WRITELN commands. This does not, however, mean than the Apple GPIB cannot be used with the Pascal System.

If you are familiar with 65Ø2 Assembly Language, and with the Apple Pascal Operating System, it is possible for you to implement most or all of the GPIB commands for use with the Pascal System.

WHAT YOU MUST DO

What you must do is write a Pascal procedure for each of the GPIB instructions that you want to implement. Some of these procedures, for example SC, can be written entirely in Pascal. Others, however, must be at least partially written in 65Ø2 assembly code. In the 65Ø2 code you send commands directly to the 9914 chip.

Before you can even attempt to write 65Ø2 routines, you must know what commands are required by the 9914. Obtain the TMS 9914 (made by Texas Instruments) specifications to find out what values must be sent to which of the registers in the 9914 to implement the GPIB instructions that you will need.

Look in Appendix C to determine the Apple addresses of the registers on the 9914. These are the addresses with which your assembly language routines must communicate. The routines that you write must communicate with the 9914 registers in the same way as the Send, Receive and Initialize routines given in Appendix E. (See the use of locations INTØ, INTMØ, INTM1, DIN and DOUT.) In fact, you can even use slightly modified versions of these three routines as the start of your Pascal GPIB package.

Now write a 65Ø2 procedure that will carry out the function that you want to implement. All that you must do is link this procedure into the Pascal system.

Information on the Apple Pascal's 65Ø2 Assembler directives may be found in the chapter on the 65Ø2 Assembler in the Apple Pascal Operating System Reference Manual (Apple Product #A2LØØ28). This chapter explains how a procedure must be structured to be linked into the Pascal System.

The chapter on the Linker in the same manual explains the actual process of linking a procedure into the Pascal System. Finally, the Apple Pascal Language Reference Manual (Apple Product #A2LØØ27) tells how EXTERNAL procedures (all assembly language procedures are external to the host program) are declared and called from within a Pascal host program.

Since your assembly language routines will be communicating with addresses that are external to the Apple, it is very advisable that you use a logic analyzer such as the ZT488 GPIB ANALYZER to monitor the signals on the bus. This may be the only way that you can verify correct bus operation.

APPENDIX G

SAMPLE PROGRAMS

This appendix contains two sample programs for the IEEE-488 card. You won't be able to use these programs unless you happen to have an HP 3438A Multimeter or an HP 3586C Selective Level Meter.

Nevertheless, they serve as excellent examples of the types of programs that can be written for the GPIB.

READ ROUTINE FOR HP 3438A

The HP3438A Multimeter has a device number of 3. Its listener address is "#" and its talk address is "C". The following program assumes that the IEEE-488 card is in slot 3. The data sent by the 3438A takes the form SD.DDDESD,F<CR><LF> for which

S is the sign, either "+" or "-"
D is a decimal digit from "Ø" to "9"
E is an "E" indicating the exponent
, is a delimiter between the value and the function code
F is the function code, a digit between "1" and "5"
<CR><LF> is carriage return followed by line feed

```
1Ø  DIM S$(3Ø): REM  ------- DELETE FOR APPLESOFT
2Ø  DIM C$(3Ø): REM   ------ DELETE FOR APPLESOFT
3Ø Z$ = "": REM     CTL-Z
4Ø D$ = "": REM     CTL-D
5Ø  PRINT D$;"PR#3"
6Ø  PRINT D$;"IN#3"
7Ø  PRINT "SC1": REM   ----- TURN THE SCREEN ON
8Ø  PRINT "TG#";Z$: REM  --- TRIGGER A READING
9Ø  PRINT "LF1": REM     ---- 3438A SENDS A <CR><LF>
1ØØ  PRINT "RDC";Z$;: REM  - READ FROM TALKER "C"
11Ø  INPUT S$,C$: REM   ---- S$ GETS CHARACTERS BEFORE
12Ø  REM   ----------------- THE "," SENT BY THE 3438A
13Ø  REM   ----------------- C$ GETS CHARACTERS AFTER THE ","
14Ø  PRINT D$;"PR#Ø": REM  - SEND OUTPUT TO SCREEN
15Ø  PRINT D$;"IN#Ø": REM  - GET INPUT FROM KEYBOARD
16Ø  PRINT "READING = ";S$;
17Ø  REM ------------------- NEXT LINE COMPUTES VALUE OF FUNCTION
18Ø  REM ------------------- CODE RETURNED BY THE 3438A
19Ø  ON  ASC ( LEFT$ (C$,1)) - 48 GOTO 2ØØ,21Ø,22Ø,23Ø,24Ø
2ØØ  PRINT " DCV": GOTO 25Ø
21Ø  PRINT " ACV": GOTO 25Ø
22Ø  PRINT " DCI": GOTO 25Ø
23Ø  PRINT " ACI": GOTO 25Ø
24Ø  PRINT " OHMS": GOTO 25Ø
25Ø  INPUT "WOULD YOU LIKE ANOTHER READING(Y OR N) ?";A$
26Ø  IF A$ = "Y" THEN  GOTO 5Ø
27Ø  END
```

GENERAL PURPOSE HP 3586C ROUTINE

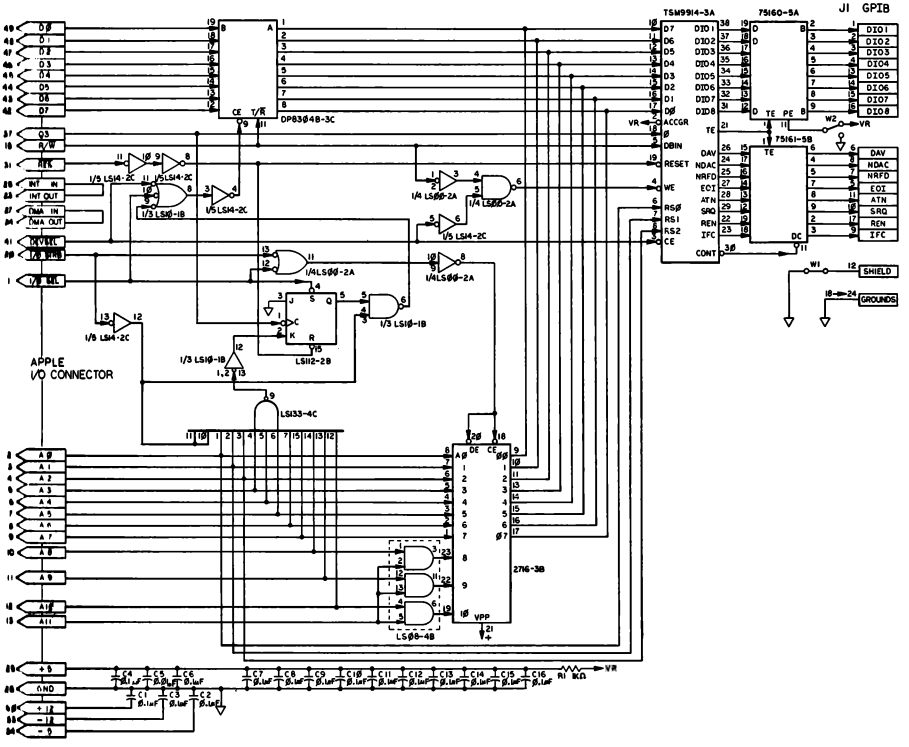
```
1Ø  REM     ROUTINE FOR HP3586C SELECTIVE LEVEL METER
2Ø  REM     METER IS DEVICE 16
3Ø  REM     LISTEN ADDRESS IS ASCII 3Ø = "Ø"
4Ø  REM     TALK ADDRESS IS ASCII 5Ø = "P"
5Ø  REM      ASSUMES CARD IN SLOT #3
6Ø D$ = "": REM    CONTROL-D
7Ø Z$ = "": REM    CONTROL-Z
8Ø  DIM C$(1ØØ): REM  DELETE FOR APPLESOFT
9Ø  GOSUB 59Ø: REM  INITIALIZE CARD FOR I/O
1ØØ  GOSUB 66Ø: REM  INITIALIZE APPLE FOR I/O
11Ø  HOME
12Ø  VTAB 5: HTAB 18: PRINT "MENU": PRINT : PRINT
13Ø  HTAB 8: PRINT "1 : PRINT TO THE DEVICE"
14Ø  HTAB 8: PRINT "2 : LISTEN TO DEVICE"
15Ø  HTAB 8: PRINT "3 : CLEAR DEVICE"
16Ø  HTAB 8: PRINT "4 : REMOTE ENABLE DEVICE"
17Ø  HTAB 8: PRINT "5 : LOCAL LOCKOUT DEVICE"
18Ø  HTAB 8: PRINT "6 : SET DEVICE TO LOCAL MODE"
19Ø  HTAB 8: PRINT "7 : ABORT SEQUENCE"
2ØØ  HTAB 8: PRINT "8 : SEND LITERAL STRING"
21Ø  HTAB 8: PRINT "9 : QUIT": PRINT : PRINT
22Ø  HTAB 8: PRINT "WHAT IS YOUR COMMAND?"
23Ø  GET A
24Ø  ON A GOTO 3ØØ,35Ø
25Ø  GOSUB 69Ø
26Ø  ON A - 2 GOTO 48Ø,49Ø,5ØØ,51Ø,52Ø
27Ø  GOSUB 66Ø
28Ø  ON A - 7 GOTO 53Ø,58Ø
29Ø  GOTO 11Ø: REM  NO SUCH COMMAND
3ØØ  PRINT "ENTER COMMAND AND DATA": REM  PRINT TO THE DEVICE
31Ø  INPUT C$
32Ø  GOSUB 69Ø: REM  INITIALIZE CARD FOR I/O
33Ø  PRINT "WTØ";Z$;C$: REM  SEND COMMAND TO METER
34Ø  GOTO 1ØØ: REM  GET NEXT COMMAND
35Ø  PRINT "INCOMING DATA IS..."
36Ø  GOSUB 69Ø: REM  INITIALIZE CARD FOR I/O
37Ø  PRINT "TGØ";Z$: REM  DEVICE MUST BE TRIGGERED BEFORE READING
38Ø  PRINT "RDP";Z$;: REM  READ FROM DEVICE
39Ø C$ = "": REM  SET C$ TO EMPTY
4ØØ  GET A$: PRINT A$;: REM  GET CHARACTER.
41Ø  REM  --- SEE NOTE FOR REINITIALIZE FUNCTION.
42Ø  IF  ASC (A$) = 13 THEN 44Ø: REM  REPEAT UNTIL <CR>
43Ø C$ = C$ + A$: GOTO 4ØØ: REM  SAVE CHARACTERS IN C$
44Ø  GOSUB 66Ø: REM  INITIALIZE APPLE FOR I/O
```

```
450  PRINT C$: REM  PRINT INPUT STRING
460  PRINT "HIT ANY KEY TO CONTINUE..."
470  GET A$: GOTO 110: REM  GET NEW COMMAND
480  PRINT "CA": GOTO 100: REM  CLEAR ALL REMOTE DEVICES
490  PRINT "RA": GOTO 100: REM  ENABLE DEVICES FOR REMOTE OPERATION
500  PRINT "LL": GOTO 100: REM  LOCAL LOCKOUT DEVICE
510  PRINT "LA": GOTO 100: REM  LOCKOUT DEVICE'S FRONT PANEL
520  PRINT "AB": GOTO 100: REM  SET ALL DEVICES TO LOCAL MODE
530  PRINT "ENTER THE LINE TO BE TRANSMITTED:"
540  INPUT C$: REM  READ LINE
550  GOSUB 690: REM  INITIALIZE CARD FOR I/O
560  PRINT C$: REM  TRANSMIT LINE
570  GOTO 100
580  END
590  PRINT D$;"PR#3": REM  SEND OUTPUT TO IEEE-488 CARD
600  PRINT D$;"IN#3": REM  GET INPUT FROM IEEE-488 CARD
610  PRINT "SC1": REM  SET THE SCREEN TO ON
620  PRINT "RA": REM  ENABLE ALL DEVICES FOR REMOTE OPERATION
630  PRINT "DV1": REM  SET THE APPLE TO DEVICE 1
640  PRINT "LF1": REM  SEND AND RECEIVE <LF> AFTER <CR>
650  RETURN
660  PRINT D$;"PR#0": REM  SEND OUTPUT TO THE SCREEN
670  PRINT D$;"IN#0": REM  GET INPUT FROM THE KEYBOARD
680  RETURN
690  PRINT
700  PRINT D$;"PR#3": REM   SEND OUTPUT TO IEEE-488 CARD
710  PRINT D$;"IN#3": REM   GET INPUT FROM IEEE-488 CARD
720  REM -AT THIS POINT ALL DEFAULT COMMANDS MUST BE RESET
730  REM -FOR EXAMPLE: LF1,SC1,DV1,ETC
740  RETURN
```

Notes: 1) Lines 410-450 contain the routine Input With Commas. Refer to Appendix D for a more complete explanation of this code.

APPENDIX H

SCHEMATIC DIAGRAM



Physical and Electrical Specifications

The Apple II IEEE-488 circuitry is mounted on a full size Apple Peripheral Card, measuring 2.75" by 7". The edge connector has 50 contacts, 25 on each side of the board. The centers of the edge connector contacts are .1 inch apart.

The card draws 400 milliamperes of current during normal operation; the maximum current it can draw is 800 milliamperes.

All of the card's outputs are three-state outputs that have the terminating resistances specified by the IEEE-488 standard.

GLOSSARY

APPLESOFT: The floating-point version of BASIC that your Apple uses. It has the CHR$ function that must be used to generate secondary addresses.

Asynchronous: Not synchronized. Synchronous events are generally clocked by the same clock pulse, asynchronous events are not.

Buffered Input: Data that is sent to a device faster than the device can process it can be stored temporarily in a special input storage area (an input buffer) and then processed by the device whenever the device is ready. Input buffers are generally internal to devices.

Control Function: A function that tells some device what to do.

CTRL: The label that appears on a key on the left side of the Apple II's keyboard. It should be held down while another key is pressed to generate the control codes in Table A-4 of Appendix A.

Controller: The device that controls all transfers of data and control bytes on the bus. The controller for the bus is the IEEE-488 card together with the Apple.

Daisy Chained: Devices that are attached to a bus one after the other are said to be daisy chained. Devices can also be star connected on the bus.

Direct Memory Access (DMA): High speed accessing of memory without the use of the CPU (in this case the 65Ø2). The Apple's processor can be suspended for DMA using pin 22 of a peripheral I/O connector slot.

Handshake: A sequence of signals transmitted over a bus that allows devices to identify themselves as ready to receive data, transmit data, or neither. The three handshake lines for any IEEE-488 bus are NRFD, DAV and NDAC.

Integer BASIC: The Apple version of BASIC that does not have the use of floating-point numbers. The CHR$ function may be simulated by using Program 1 from Appendix D.

Interface: As used in this manual, the hardware and software that allows two devices to communicate.

Jumper: One or more wires (sometimes made into a plug) connected in one of several alternative ways. Just as you can use jumpers to set the memory configuration of the Apple, manufacturers often use jumpers to set the address to which a device responds.

Listener: A device that is enabled to receive data over a bus.

Local Operation: Operation of a device using the knobs and switches on the device front panel, as opposed to remote operation on the GPIB.

Parallel Poll: A method of simultaneously asking many (for GPIB, up to eight) devices if they are waiting to be serviced. A device that is waiting to be serviced will respond to the poll by setting a bit in the status byte read by the PP command.

Piggyback Connectors: Connectors that have plugs above their pins, allowing more than one cable to be connected at the same point.

Remote Operation: Operation of a device using control information device (for example, the controller on the GPIB).

Serial Poll: A method of polling that asks each device in turn if it needs servicing. This method is slower than parallel polling but it can be used with more devices. A serial poll is initiated after the SRQ line is pulled low by a device, and it uses the SRQ command.

Standard: A way of doing something that has been defined by a standards committee. Products that adhere to a single standard are compatible with each other. For example, the Apple GPIB adheres to the IEEE-488 standard.

Star Connected: A bus configuration in which one device is like the hub of a wheel with the other devices on the rim of the wheel, connected to the first device along the spokes. This configuration is not recommended.

Talker: A device that is enabled to transmit data over a bus. The IEEE-488 standard permits only one talker at a time.

Term: Short for terminator.

Terminator: A particular symbol or character that is used to mark the end of something (for example, a string of data). <CR>, <EOS> and EOI are all used as GPIB terminators.

TMS 9914 - The GPIB adaptor chip around which the Apple GPIB is designed. It is made by Texas Instruments.

Trigger: To cause something to happen immediately. Triggers are used for synchronization or enabling events.

INDEX

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

APPLE GPIB COMMAND SUMMARY

COMMAND	FORMAT	BASIC	BUS?	PAGE
WRITE	WT <listen list> <data> <EOS>	PRINT	YES	29
WRITECNT	WC <listen list> <count> <data>	PRINT	YES	30
READ	RD <talker> <CTRL-Z> <data> <term>	INPUT	YES	30
READCNT	RC <talker> <CTRL-Z> <count> <data> <term>	INPUT	YES	31
XFER	XF <talker> <CTRL-Z> <listen list> <CR>	PRINT	YES	32
TRIGR	TG <listen list> <CR>	PRINT	YES	32
CLRAL	CA <CR>	PRINT	YES	32
CLEAR	CL <listen list> <CR>	PRINT	YES	33
REMAL	RA <CR>	PRINT	YES	33
REMDV	RM <listen list> <CR>	PRINT	YES	33
LLKAL	LL <CR>	PRINT	YES	34
LOCAL	LA <CR>	PRINT	YES	34
LOCDV	LO <listen list> <CR>	PRINT	YES	34
SRQD	SR <srq> <CR>	INPUT	YES	35
SPOLL	SP <talker> <CTRL-Z> <status word> <CR>	INPUT	YES	35
PPOLL	PP <status word> <CR>	INPUT	YES	36
PPENB	PE <enable list> <CR>	PRINT	YES	36
PPDIS	PD <listen list> <CR>	PRINT	YES	37
PPUAL	PU <CR>	PRINT	YES	37
ABORT	AB <CR>	PRINT	YES	37
LINEFEED	LF <off/on> <CR>	PRINT	NO	27
EOS	ES <EOS> <CR>	PRINT	NO	28
SCREEN	SC <off/on> <CR>	PRINT	NO	28
DEVICE	DV <device number>	PRINT	NO	29
UNTALK	UT <CR>	PRINT	YES	37

APPLE GPIB COMMAND STRINGS

<listen list> ::= <CTRL-Z> | <listener> <CTRL-Z> | <listener> <listen list>

<listener> ::= <primary listener> | <primary listener> <secondary address>

<primary listener> ::= <SP> | ! | " | # ... > | ?

<secondary address> ::= ' | a | b | c ... } | ~

<count> ::= <decimal digit> <CTRL-Z> | <decimal digit> <count>
NOTE: The maximum total value for count is 255.

<EOS> ::= any ASCII character

<data> ::= empty | any ASCII character | any ASCII character <data>

<talker> ::= empty | <primary talker> | <primary talker> <secondary address>

<primary talker> ::= @ | A | B | C ... ^ | _

<hex digit> ::= 0 | 1 | 2 ... 9 | A | B ... F

<decimal digit> ::= 0 | 1 | 2 ... 9

<status word> ::= <hex digit> <hex digit>

<srq> ::= ASCII "T" | ASCII "F"

<term> ::= empty | <CR> | <EOS> | EOI
NOTE: empty if last char = <CR> or <CR><LF>

<device number> ::= <decimal digit> <decimal digit>
NOTE: Maximum device number is 30

<off/on> ::= ASCII "0" | ASCII "1"

<enable list> ::= <listener> <enable> | <listener> <enable> <enable list>

<enable> ::= @ | A | B ... N | O

<CR> ::= ASCII carriage return (13 decimal)

<LF> ::= ASCII linefeed (10 decimal)

<SP> ::= ASCII space (32 decimal)

<CTRL-Z> ::= ASCII SUB (26 decimal)

<CTRL-D> ::= ASCII ET (4 decimal)

* NOTE: Marked elements are ASCII characters.

10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

030-0197-A